AFRL-IF-RS-TR-2006-119
Final Technical Report
April 2006

JOINT EXPERIMENTATION ON SCALABLE PARALLEL PROCESSORS (JESPP)

University of Southern California

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-119 has been reviewed and is approved for publication

APPROVED: /s/

DUANE GILMOUR
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS
Deputy Chief, Advanced Computing Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	APRIL 2006	Final Sep 02 – Sep 04

4. TITLE AND SUBTITLE
JOINT EXPERIMENTATION ON SCALABLE PARALLEL PROCESSORS (JESPP)

5. FUNDING NUMBERS
C - F30602-02-C-0213
PE - N/A
PR - JESP
TA - PS
WU - 02

6. AUTHOR(S)
Dan M. Davis, Robert F. Lucas, Ke-Thia Yao, Gene Wagenbreth

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
4647 Admiralty Way
Marina Del Rey, California 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFTC
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-119

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Duane Gilmour/IFTC/(315) 330-3550 Duane.Gilmour@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
The Joint Experimentation on Scalable Parallel Processors (JESPP) project exemplified the accessibility and the utility of High Performance Computing for large-scale simulations. In order to simulate the future battlespace, the US Joint Forces Command's (USJFCOM) Experimentation Directorate (J9) required expansion of its joint semi-automated forces (JSAF) code capabilities; including number of entities, behavior complexity, terrain resolution, infrastructure features, environmental realism, and analytical potential. The USJFCOM J9 was charged with developing a very large-scale simulation capability of future combat environments, particularly urban areas, with more than one million civilian simulated entities. Synthetic forces have long run in parallel on networked computers. The JESPP strategy exploited the scalable parallel processors (SPPs) of the High Performance Computing Modernization Program (HPCMP). SPPs provide a large number of processors, interconnected with a high performance switch and a collective job management framework. To achieve the goal of simulating one million entities, software routers were developed that replaced multicast with point-to-point transmission of interest-managed packets. This final report lays out that design and development. It also details several experimentation events that have simulated up to one million clutter entities, which were "fought" from Suffolk, VA.

14. SUBJECT TERMS
Large scale simulation, semi-automated force simulation, distributed supercomputer, software routers, scalable parallel processor supercomputing

15. NUMBER OF PAGES
105

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1.0 Executive Summary

The Joint Experimentation on Scalable Parallel Processors (JESPP) project exemplified the accessibility and the utility of High Performance Computing for large-scale simulations. In order to simulate the future battlespace, the US Joint Forces Command's (USJFCOM) Experimentation Directorate (J9) required expansion of its joint semi-automated forces (JSAF) code capabilities; including number of entities, behavior complexity, terrain resolution, infrastructure features, environmental realism, and analytical potential. The USJFCOM J9 was charged with developing a very large-scale simulation capability of future combat environments, particularly urban areas, with more than one million civilian simulated entities. Synthetic forces have long run in parallel on networked computers. The JESPP strategy exploited the scalable parallel processors (SPPs) of the High Performance Computing Modernization Program (HPCMP). SPPs provide a large number of processors, interconnected with a high performance switch and a collective job management framework. To achieve the goal of simulating one million entities, software routers were developed that replaced multicast with point-to-point transmission of interest-managed packets. This final report lays out that design and development. It also details several experimentation events that have simulated up to one million clutter entities, which were "fought" from Suffolk, VA. These entities were typically executed on remote SPP systems, one in Maui, Hawaii and one in Dayton, Ohio. This report further sets forth the experience in scoping the high performance computing hardware needs to support SPPs, developing the project with the HPCMP, and implementing the system.

2.0 Methods, Assumptions and Procedures

The long-term objective of the USJFCOM has been to lead the transformation of the United States Armed Forces to achieve full-spectrum dominance, as described in Joint Vision 2010 and 2020. The research arm of the USJFCOM is J9, which integrates experimentation efforts of the services and unified commands. It has been and is America's military transformation laboratory. To meet the DoD transformation goals, USJFCOM conducts exercises of increasing size, capability and resolution. These increases needed to be orders of magnitude larger than those previously possible.

To accomplish an exercise of this unprecedented scale, the first and most obvious task was the implementation of the core of the JSAF program on a platform capable of supporting terascale computing: the SPP environment. This was the foundation for this effort. This implementation could only be effective with the best parallel architecture possible, using the best practices of parallel programming and system engineering. Some of this work had commenced under a previously conducted effort, funded by the Defense Advanced Research Projects Agency (DARPA). This was the well-founded basis on which to build a program for this effort. In addition, several tasks had to be completed and tools provided to enable the exercises of increasing size, capability and resolution. These tasks include:

- Clutter definition and SPP adaptation
- Terrain server implementation on SPP nodes

- Terrain data base definition and SPP adaptation
- Exercise initiation utility creation (to lay down entities)

To exploit and verify SPP capabilities in this field, the initial focus was on an evaluation in which one million clutter entities were to be exercised on a large-scale, high resolution terrain database (TDB).

In order to simulate the future battlespace, the USJFCOM J9 had to expand the capabilities of its JSAF code along several critical axes; including continuous experimentation, number of entities, behavior complexity, terrain databases, dynamic infrastructure representations, environmental models, and analytical capabilities. Increasing the size and complexity of exercises supported by JSAF, in turn, required increasing the computing resources available to the USJFCOM. The approach pursued in this effort was to exploit SPPs deployed by the DoD's HPCMP. Synthetic forces had long run in parallel on networked computers. SPPs were a natural extension of this, providing a large number of processors, interconnected with a high performance switch, and a collective job management framework. To effectively use an SPP, software routers that replace multicast messaging with point-to-point transmission of interest-managed packets were developed. This in turn required development of a new simulation preparation utility to define the communication topology and initialize the exercise. Tools were developed to monitor processor and network loading, as well as loggers capable of absorbing all of the exercise data.

Current and future operational imperatives are driving experimental designs which require further expansions of JSAF capabilities. As noted before, some of the requirements justifying these extensions were the need for:

- More entities
- More complex behaviors
- Larger geographic area
- Multiple resolution terrain
- More complex environments

The most readily available source of one or more orders of magnitude of increased compute power was the capability presented by SPPs. In this project, the JSAF code was ported to run on multiple Linux clusters, using hundreds of processors on each cluster. Future runs will require thousands of processors on multiple clusters. The primary difficulty in using these resources was the scaling of internode communication.

The User Datagram Protocol (UDP) multicast was limited to approximately three thousand different channels. Based on geography alone, worldwide simulations using JSAF require many more interest states. To enable this, the UDP multicast had to be replaced by software routers.

Software routers were implemented on individual nodes in a network that included all of the client simulators. Each simulator was connected to only one router. Routers were connected to multiple clients and multiple routers. Each connection was a two-way connection. Two types of

information were present in the network. One was data from the simulation engines along with interest descriptions. The other was the current interest state of each client. The interest state changes as each node subscribes and unsubscribes to specific interest sets, as was appropriate depending on the simulation progress.

Each router had to maintain the interest set of each node to which it was connected, including other routers. A router's interest set was the union of all the connected nodes. A router then used the interest state associated with the data it receives to determine how to forward the data. For a given topology, communication was minimized such that each client node received exactly the data in which it was interested.

The initial router implementation was a tree router. Each router had multiple clients but only one parent. There was one router that was at the top of the tree. A second topology was subsequently implemented. This is referred to as a mesh router. Instead of a single router at the top of a tree, there was a mesh of routers with all-to-all communication. Each simulator was a client of one of the mesh routers. Like the tree router, the primary task of the mesh router was to maintain the interest state of all clients so as to forward only data that was of interest to each client and router. Further hybrid topologies were made possible with little or no code modification, such as a mesh of meshes or a mesh of trees. Conceptually, the mesh provides better scalability, and in practice this has been demonstrated.

Another use of routers was the implementation of gateways providing an interface between different runtime infrastructure (RTI) and communication implementations. Both transmission control protocol (TCP) and UDP were used for communication. Routers could use a different protocol on different connections and perform required data bundling, unbundling, etc. Different RTI implementations, required by simulators developed by different groups, could communicate via router-based gateways. A router design diagram is shown in Figure 1.

The ultimate goal was for the capacity of a simulator network to scale easily as the numbers of processors were increased by several orders of magnitude. Comprehensive testing and measurement was required to document the performance of various topologies and router implementations. This testing identified performance bottlenecks and suggests alternative implementations to be tested. Multiple simulation scenarios are required to be tested to construct guidelines for assigning simulators, routers and topologies to multiple SPPs.

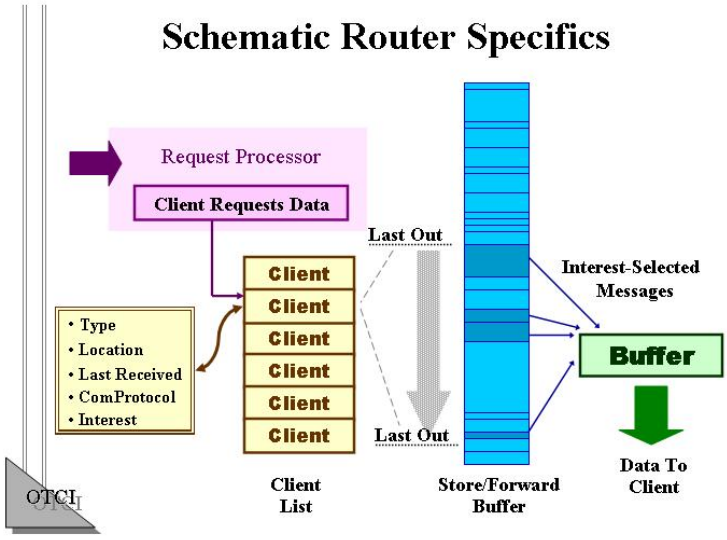


Figure 1 - Router Design Diagram

To demonstrate SPP capabilities, as well as to support future use and development of joint experimentation, clutter entities were identified and defined. Further, this allowed for the identification, investigation, modification, creation and validation of a scalable parallelized terrain server for use in SPP simulations using JSAF. To enable this server development and test, a terrain database was selected, analyzed and modified, as required, for parallel use on SPPs and prepared for use in the test. In addition, to enable rapid testing without significant operator involvement, a utility to lay down the entities in a realistic and efficient manner was developed and incorporated into early and subsequent tests.

3.0 Project Goals

The following specific tasks were identified as necessary in pursuit of the project goals:

- Parallelize critical JESPP functions
- Operationalize the JESPP capability
- Implement JESPP on two HPCMP SPPs via the Defense Research and Engineering Network (DREN)
- Improve control, performance and behaviors
- Emphasize urban environments
- Participate in spiral events, experiments and tests
- Implement and verify urban capabilities
- Implement a system to log and analyze appropriate data
- Continue to develop fault tolerant operations
- Assist Toyon Corporation in implementing the Simulation of the Locations and Attack of Mobile Enemy Missiles (SLAMEM^{TM}) capability on the cluster

- Install and coordinate the new J9 HPCMP Distributed Center (DC) Cluster
- Support communications, security and encryption

3.1 Results and Discussion

The initial task for this project focused on successfully implementing a scalable computational environment for JSAF. This was followed by efforts related to fault tolerance, experiment initialization/entity control, multi-platform portability, and data logging. All of these issues were driven by the implementation of the SPP technology. Appropriate centers of expertise were identified to resolve each of these critical areas in a way that allowed early utilization of the SPP technology in Joint Experimentation and produced durable and robust solutions that withstood the rigor imposed by the ever-changing needs of USJFCOM.

Throughout this work, the following fundamental design goals were pursued:

- Interest management and communication must be scalable.
- There could be no artificial barriers to the number of entities allowed.
- There must be minimal imposition on the JSAF source code.
- A capability to introduce new modules containing undefined sensors and other novel entities must be enabled.
- J9's computational power must continue to be increased to enable it to represent new features; such as communication systems, dynamic terrain and advanced weather models.

Details of the principle tasks that were identified by the USJFCOM J9 follow. Numerous technical papers were written during the course of this project and are included in the Appendix and should be referenced in support of the results that follow.

3.1.1 Parallelize critical JESPP functions

The enabling of the preliminary use of the computing power that was provided by parallel computers was accomplished in three phases. They were:

- Implement parallel routers in MPI
- Design and code a socket programmed version
- Test, operate, evaluate and update code

The first phase was accomplished early in the effort, just after the commencement of this project. Two versions of the routers were implemented: the tree routers and the mesh routers. A notional diagram appears in Figures 2 & 3 and the papers by Dr. Gottschalk in the Appendix provide more explicit descriptions of the differences and relative advantages of the two designs.

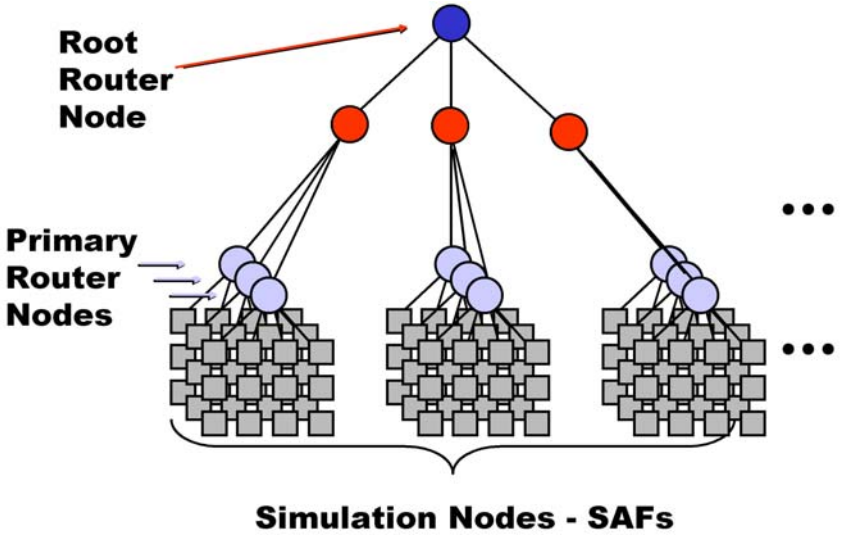


Figure 2 - Notional Diagram of Tree Router Implementation

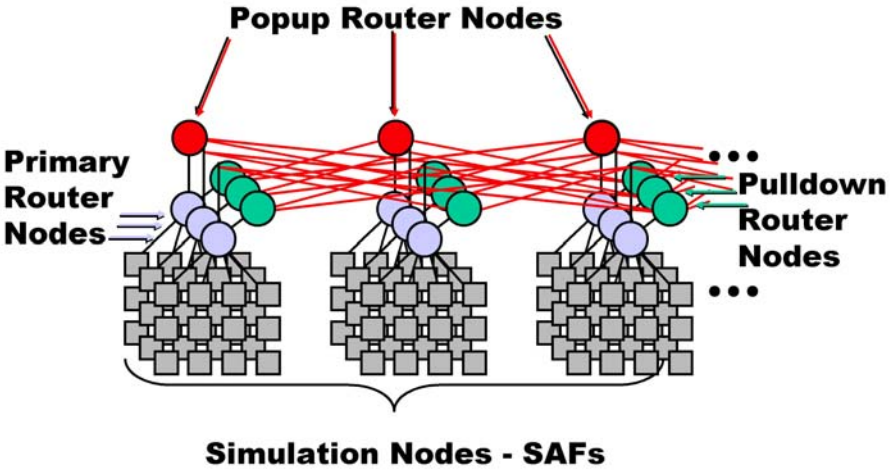


Figure 3 - Notional Diagram of Mesh Router Implementation

As there are many instantiations of modules on each node, ranging from several hundred to several thousand, the most critical function to parallelize was the internode communications. These communications were required when one entity needed to send its location, orientation and state to another entity. To avoid the "n squared" scaling typical of all-to-all communications, filtering at the router level restricts traffic flow to the lowest acceptable levels. Interest management enables a scalability that permits entities within relevant proximity to communicate, while obviating useless communication between entities so geographically separated that the data is superfluous, *e.g*. a tank in Baghdad has no great need for real-time updates on a tank in Basrah. The difficulties come, obviously, when you have a high-altitude sensor that can see all of the entities in both cities.

Initial testing established that the tree routers provided acceptable scalability and performance for the level of simulations being run as part of the JFCOM experiments. As the tree was more intuitive, it was used by the operations personnel. As noted in the papers in the Appendix, the tree routers are less scalable than the mesh routers. That suggests a future adoption of the mesh routers as the more scalable standard, but that move was not implemented in this period of performance.

Another important function that was amenable to parallelization was the data logging requirement. As the computation, data and the users are trans-continentally distributed, a new method of data logging was required. Working with scientists from the Institute for Defense Analyses, a data logger was designed that was easily inserted into the JSAF code. This logger was based around the intercept of data when it was originally communicated. This process is more fully outlined in the papers by Dr. Yao, all of which can be found in the Appendix. Figure 4 is a flow-chart representation of this design.

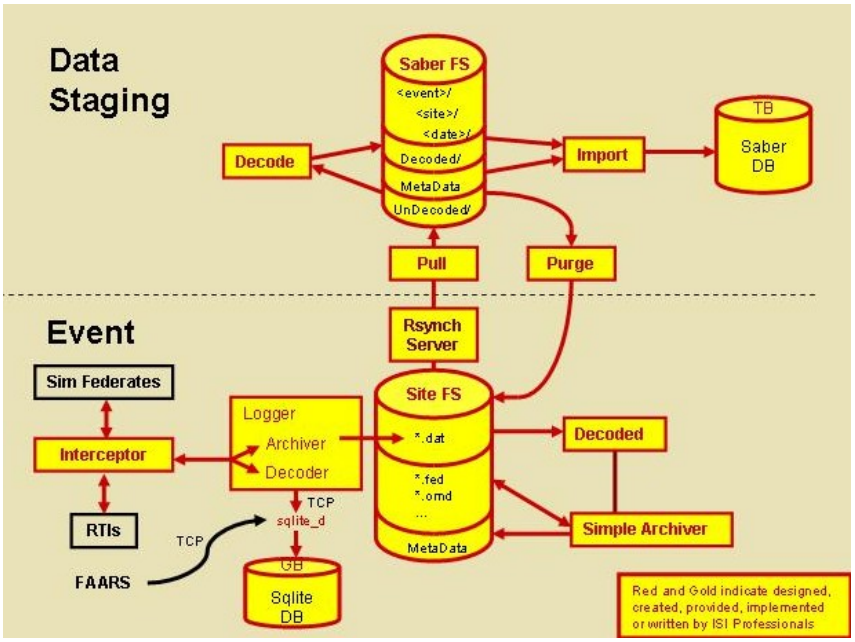


Figure 4 - Flow Chart Diagram of Data Management

Operations conducted using this system showed adequate performance and the system proved effective at collecting and managing up to two terabytes of data per week. It should be noted that this amount of data is as much as three orders of magnitude less than the final data flow anticipated if the program were to log all data, increase the sophistication of the behaviors and increase the number of entities. All of these increases are ardently sought by the Joint Forces Command. The data management system shows no natural scalability constraints and the principal reason for not logging all of this data is that the physical devices for storage, *i.e.* a disk array, is limited to two terabytes.

Testing continued for the rest of the period of performance, with significant gains in stability and performance being indicated. The tree routers exhibited a natural constriction at the root node, which led to higher latency than necessary; redundant circuit paths *e.g.* routing messages from Maui to Norfolk and then back to San Diego; and an obvious "single point of failure." Continued testing of mesh router performance demonstrated the desirability of implementing that design.

Figure 5 below gives an indication of some of the performance results obtained during the research on this project. This chart was generated using a test code that fired off successions of messages of varying sizes. The vertical axis represents the result of dividing the tree router times by the mesh router times for comparable conditions. While scalability rather than throughput was the goal, a performance gain shows a good foundation for future scalability. Number of "hops" refers to how many routers intervene between nodes used in this test.

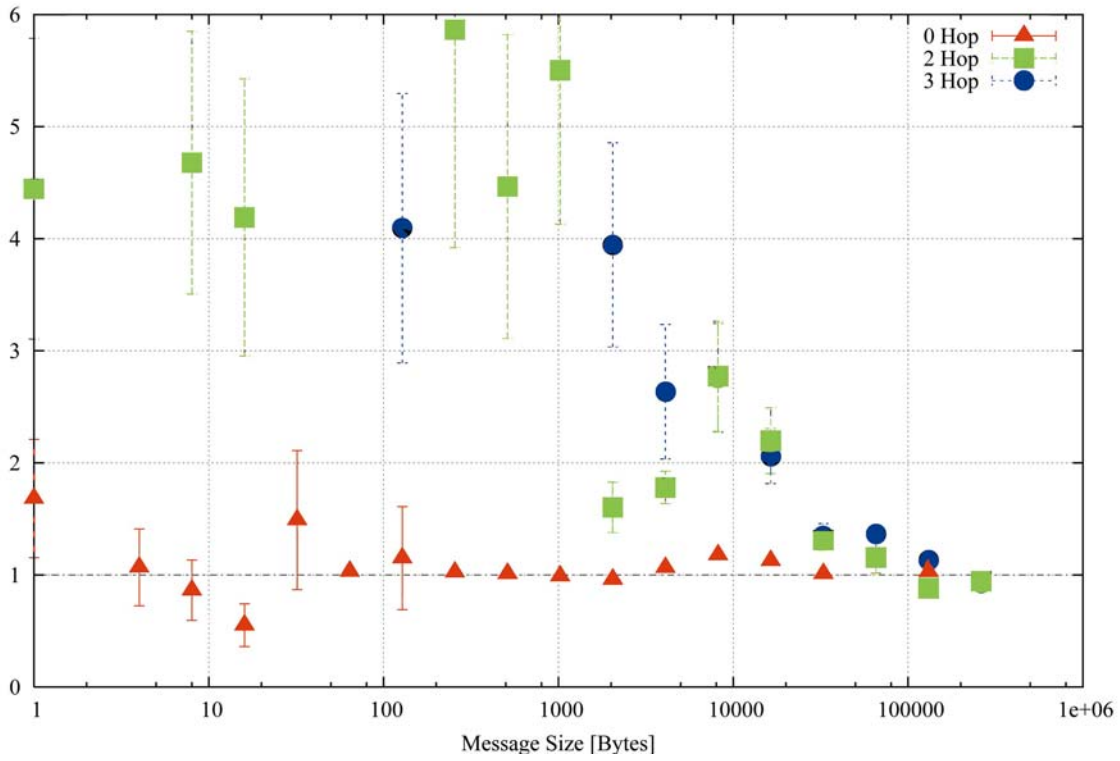


Figure 5 - Semi-log Chart of Ratio of Performance Advantage of Mesh over Tree Routers using Test Message Inter-node Communications

3.1.2 Operationalize the JESPP capability

The goal of this task was operationalizing the JESPP JSAF code. This required enabling it to tolerate failure in any individual processor in the SPP. There could be no single point of failure. Among other things, this would allow the user to stop and restart any arbitrary node on the SPP, including the routers. Several concepts for fault tolerance were evaluated:

- Redundant, stand-by nodes
- Mirrored router nodes
- Dynamic, adjustable mesh to optimize entity densities
- Use of new fault tolerant communication fabric for the mesh router
- Utilization of Globus techniques to switch tasks between pre-configured partitions of SPPs

The principle task under this segment was the provision of providing an operationally stable and usable system. This single effort was of the greatest concern to the Joint Forces Command and was the most importane advance required by the Joint Experimentation group.

A reprioritization of this task was necessitated by emerging issues not well-recognized or defined prior to the execution of the contract, e.g. fault tolerance was not nearly the concern that was anticipated, while security issues became more of a driver for the system development. There was a constant pressure and continuous effort to further refine the newly parallelized, scalable code. The operational/development paradigm for JSAF in the USJFCOM implementation is to have a real-time, interactive development during the experiments. For instance, during any experiment, changes to the code could be anticipated several times a day for two weeks. These changes required a rapid analysis by JESPP computational scientist and an assiduous attendance to detail and attendance at code development meetings and experiment planning sessions. This illuminated the need for a deep understanding of the subject code and a close and collegial working relationship with programmers and simulation operators. This allowed for many unheralded and rapid responses to changes being implemented. In general, fault tolerance at the node or partition level was not a major issue.

Operational issues that were faced were far more significant than fault tolerance. As an example, during this effort, SPP assets were not available for two principle reasons, center downtime for major upgrades or security operations or center off-line status due to power or communications failures. There were three major outages, maintenance took Aeronautical Systems Center Major Shared Resource Center (ASC MSRC) off-line, a storm disrupted communications at JFCOM, and a power outage on the island of Maui took down the Maui High Performance Computing Center (MHPCC). In none of these cases, would a mesh redundancy or fault tolerance been of any significant use.

One of the major contributions of this effort was the path-finding nature of establishing real interactive operations at the HPC level. The research community is largely operating in the batch mode, but several individuals are leading efforts to enable and advocate for interactive HPC. As the JESPP project progressed, it became obvious that it was not only the groundbreaking project, but, due to its high visibility, it was used as an example of the need for such a computational environment.

A significant amount of time was expended developing close coordination with the other members of the JSAF team, especially Lockheed Martin Information Systems (LMIS). The experimentation environment provided a continuous testing and evaluation setting for exercising

the SPP environment. All modifications were carefully coordinated with JSAF team, with an eye toward not making changes that would hamper their use in ongoing exercises. The JESPP team also designed their code submissions to avoid mandating modifications that would have imposed onerous re-coding of existing modules.

3.1.3 Implement JESPP on two HPCMP SPPs via DREN

This task focused on implementing the current and future capabilities on high performance hardware owned and controlled by the DoD. Close attention was paid to utilizing HPCMP computers, if possible. Suitable SPPs were identified and selected, which were comparable to the Linux clusters used in the initial development of JESPP. An investigation into the accessibility and range of SPP computers available to J9 experimenters at the HPCMP sites was accomplished. Our personnel were familiar with current trends in SPP computing and computational science initiatives and used that as background as they assiduously identified and documented the planned future platforms at the HPCMP sites. Having established the likely range of SPPs to be available in the future, the team carefully assessed the needed modifications to the JESPP system to ensure easy portability, hardware and software compatibility, and incorporation of enhancements enabled by the projected future advances. Having done so, the team picked a few representative SPPs, and conducted a series of portability tests and performance evaluations. Two computers were arranged to support a prototype event utilizing the J9 experimental test bays. These machines had DREN connectivity which was sufficient to provide bandwidth on the order of 50 Mb/s to the JESPP. Our personnel assisted J9 in designing, selecting, installing and initializing a "Beowulf" Linux cluster at J9, which was initially comprised of 16 dual-processor nodes.

3.1.4 Improve control, performance and behaviors

This task focused on making the conceptualization, definition, and initialization of each experiment more accessible to experimenters. This was one of the major goals set forth in two workshops on Joint Experimentation on SPPs in 2002. The process was both intuitive to the new experimenter and familiar to the experienced J9 personnel. The JESPP team had to learn the JFCOM culture of code improvement and modification. As this is a rather unusual process, a few words might help explicate the issue.

While literally volumes have been written on the standard code development paradigm, the JFCOM system responds of necessity to a different operational tempo. The standard environment envisions a batch operation with fixed deadlines, clearly delineated ahead of time, as follows: "we must double the capacity of our buffers by this day, all organizational rules for development, test and documentation are in force." The JFCOM scenario is much more likely to be, "the experiment director (not infrequently a flag officer) liked the run this morning, but after lunch he wants the entities to display emotional characteristics and the terrain to show impact results from weapons." As there are up to one hundred participants in a typical experiment, taking the time to carefully follow coding conventions, including adequate documentation, would cost not an hour per programmer hour, but 100 staff hours per each hour the experiment waited on coding changes.

This incredibly interactive programming style had two major impacts on the JESPP team.

- The C++ code with which they had to interface was very dynamic and bore constant monitoring to keep the routers and high performance computing from being the failure point for the experiment
- The body of the code itself, some two and half million lines of it, was largely un-documented as far as avoiding pitfalls

As an example of how this impacted the crew, there was an occasion where the JFCOM developers had inserted a piece of code to explicitly identify a "hard-wired" circuit for inter-node communications. This change, as was the tradition, was known only to those working the problem at that time. When the mesh routers were applied to this code, performance dropped dramatically. The mesh routers were automatically establishing links that were not only redundant to, but were actually stimulating the new code to spawn several links for each communications path. This, naturally, adversely impacted performance. Our team was able to overcome these issues and become productive in the demanding short time periods available to implement on-the-fly fixes.

The approach built on previous tools and procedures, including Multisystem Automation Remote Control and Instrumentation (MARCI). The goal of this task was to approach the optimal distribution of the various computational and visualization tasks to the most efficient assets available. It was conceivable that all computation and visualization could be done on the mesh of the high performance computers, with only the X Windows screen data being passed over the network

3.1.5 Emphasize urban environments

This task entailed the on-going analysis and support for the continuing improvement of the JESPP's ability to support experimentation in urban environments. This was manifest both in the high-fidelity urban terrain databases and in the inter-visibility issues raised when combatants are active in a congested area. This task was integral with all of the other tasks enumerated herein, in that each activity was oriented toward improving the fidelity, utility, and validity of JSAF models in an urban setting. Inclusions of other federates, improved terrain mapping of man-made infrastructures, better representations of buildings, and increased entity densities were all necessary to support urban environments. Figure 6 shows a screen capture of a typical urban terrain environment.

Figure 6 - Screen Capture of Typical Urban Terrain Environment

3.1.6 Participate in spiral events, experiments and tests

An event was designed, planned, organized, and conducted in order to demonstrate code improvements and to incorporate as many of the capabilities listed above into the JSAF code base. Specific goals, dates, locations, hardware, network and participants were suggested by our team and approved by the leadership.

The schedule was typically a spiral development event every month, usually two weeks in duration, to prepare for the main experiment. This required the involvement of both development activities and preparation for on-line consulting and trouble-shooting during the "record runs" that were the ultimate goal and final punctuation of the spiral events. During this preparation, new code was developed, tested and submitted to the concurrent versions system (CVS) tree maintained by Lockheed Martin. On the weeks of the runs themselves, JESPP personnel were present at numerous locations to see first-hand how the test was going as well as to perform consulting, trouble-shooting and support activities.

3.1.7 Implement and verify urban capabilities

This task focused on implementing and verifying the urban environment capabilities developed in the previous tasks. Those capabilities that were ready to be utilized as part of the USJFCOM Distributed Continuous Experimentation Environment (DCEE) base capability were used to support a J9 human-in-the-loop project.

This implementation involved the following three characteristics:

- A global-scale, low resolution terrain upon which to set the local action
- A high resolution inset of a data set representing an urban area
- A believable set of civilian clutter distributed in that environment

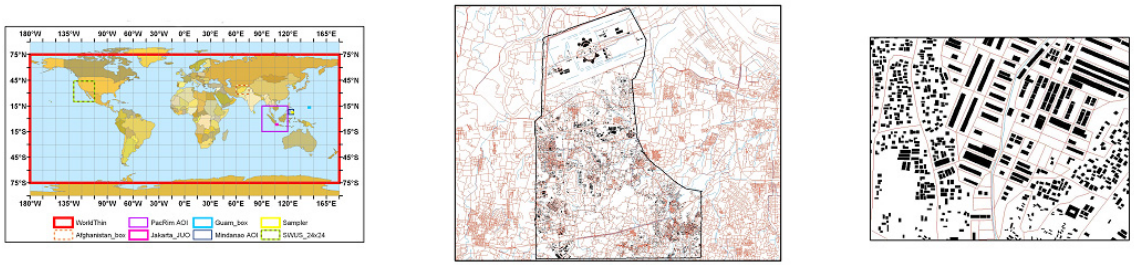


Figure 7 - Global, Area of Interest Specific, and Buildings in Variable Resolution Terrain Databases

The JESPP team provided scalable computing support for these three objectives. This would entail simulating the previously fielded civilian entity group several times. It also mandated that the terrain and the simulated entities would be distributed effectively across the nodes and would scale appropriately. This was accomplished and the requisite runs were supported. This was successfully accomplished and utilized in the experiments. Calls by individual nodes on the terrain database did not overtax inter-node communications and performance as was expected. In this, as well as other cases, the most critical performance measure was the perceived acceptability by the users, operators, analysts, and managers of the experiment.

Having successfully implemented and tested a scalable computational environment for JSAF, we contributed to the joint urban operation human-in-the-loop (JUO-HITL) spiral development. We identified the appropriate centers of expertise to resolve each of the critical areas in a way that allowed early implementation of SPP technology in Joint Experimentation. The continuing challenge was to produce durable and robust solutions that stood the rigor imposed by the ever changing needs of the USJFCOM. New capabilities in experiment initiation, entity control, data logging and after action analysis were identified and were pursued, as described below.

3.1.8 Implement a system to log and analyze appropriate data

This task focused on implementing distributed and collated data logging and storage of experimental results. The best technical description of this work can be found in the papers authored by Dr. Yao in the Appendix. Developing collaborative planning and decision support systems designed to establish user situational awareness present an additional challenge because

of the reliance on human subjects as integral components of the command and control system. Immersion of humans within virtual simulations, such as JSAF, requires an integrated data generation and collection approach to achieve quantifiable results.

One of the initial decisions made by the JFCOM data management staff was to select Microsoft Access2000® as the intermediate data store since it was already designed as the *hlaResults* primary database. The team's ability to rapidly prototype the Future After-Action Report System (FAARS) during early development was facilitated by this decision. However, as the spiral development process moved forward, new requirements surfaced for increasing the size of the data storage capability for reasons explained below. The emerging result was a continued improvement to the capabilities of the FAARS as it is adaptable to a variety of new Commercial Off-The-Shelf (COTS) database products.

While Microsoft® has produced an easily mastered product for home and small office use, the created databases within Access2000® are limited to two gigabytes. Experience showed that a typical HLA federation with 35,000 entities will quickly overload an Access2000® database, created using the utility *hlaResults*. It will reach this threshold within two to two and a half hours of event runtime. A typical simulation day consists of 6 to 8 hours of continuous runtime, requiring three or more databases to be created. A technique for selecting and processing relevant data from each database (and taking into account overlaps within the data) was developed by inserting the data into a much more richly capable database, *MySQL*, for a "roll-up" into one total event period, to preclude retrieving data separately from each Access2000® database. Subsequently, two *hlaResults* collectors were used during experiments in overlapping periods so that continuous coverage of simulation data would be recorded.

Two major benefits accrued from the use of *MySQL* and *sqlite*. First, as they are obtainable as open source, they can be obtained and distributed without cost or significant administrative burden. Secondly, as open source, the code itself can be examined for performance enhancement opportunities, customized code insertions and security robustness. While user interfaces may be marginally more austere, they are invariably accessible by technically trained personnel likely to use them.

The distributed logger made use of local embedded relational databases, implemented using *sqlite* on each node of an SPP to execute queries and return results via an ad hoc protocol implemented by a tree of aggregators. This work improved performance and provided additional functionality. Performance was improved by using *MySQL* and *PostgreSQL* databases in lieu of *sqlite* and through the use of data compression in the aggregators. Functionality was enhanced by defining a higher-level portable protocol in the aggregators, by providing standard interfaces for combining query results returned by multiple SPP nodes, and by providing methods to easily redistribute data to a different topology of nodes. Tools were developed and implemented to assist the experimenter in identifying, recovering and understanding the data that was stored.

A notional flow chart of the developed system shows important features and the complexity of the code developed to serve the needs of the users and analysts at JFCOM.

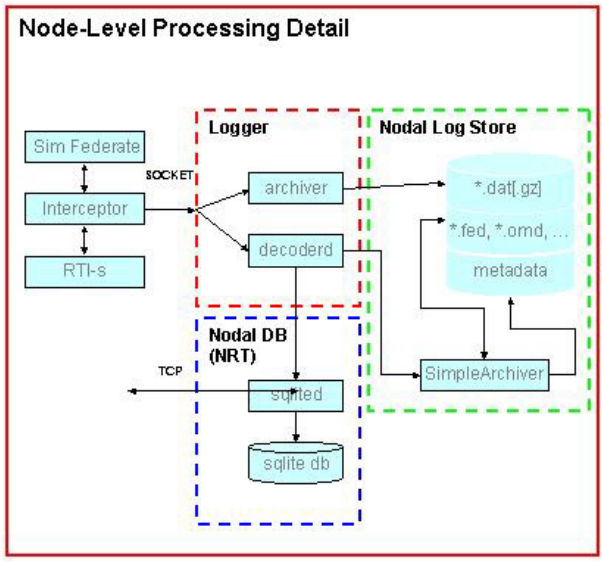


Figure 7 - Flow Chart of Data Logger System

Using this system, the Data Logger consistently logged one terabyte of exercise data each week of an experiment. While this still required discarding most of the location, orientation and state data for the clutter entities, it still represented a scalable approach, more limited by hardware costs than design constraints. This terabyte reflected approximately a two order of magnitude increase from the limits imposed by the previous system.

3.1.9 Continue to develop fault tolerant operations

The SPP implementation of JSAF was increasingly made fault tolerant and confirmed by additional study and testing. One of the most glaring of the fault susceptible features was the continued use of the tree routers. This resulted in all of the experimental data being run through a single computer. The implementation of the mesh routers in the future should dramatically reduce that risk. Critical aspects of the final implementation, based on the mesh router system, included:

- A scalable interest management system
- No artificial barriers to the number of entities allowed
- Fault tolerance:
    - No single point of failure
    - Dynamic addition/deletion of nodes
    - Early investigation of migration of entities across nodes

3.1.10 Implement SLAMEM on a cluster

The SLAMEM code was previously being operated on any number of single processor platforms. Some initial studies on the appropriate parallelization of the SLAMEM code were conducted and assistance was provided to the Toyon Corporation in efforts to make use of the scalable capabilities of the Linux clusters. The JESPP team did an external review of the

SLAMEM characteristics and identified several areas where the needs of the user seemed to suggest a new program that would more accurately and dynamically model the sensors. Specifically, the use of partial differential equation generated physics rather than subjectively generated "look-up tables" would significantly improve both the responsiveness and validity of the simulation. This work would be beneficial to the JSAF simulation and is suggested as a potential follow-on effort.

3.1.11 Install and coordinate the new J9 HPCMP Distributed Center (DC) Cluster

Our team led the effort, which resulted in the award of a DC cluster to the USJFCOM by the High Performance Computing Modernization Program (HPCMP). This important new USJFCOM asset required significant coordination to maximize its utility to the Joint Experimentation group. Initially, there was a need to expend significant amounts of time in acceptance testing and coordination to satisfy HPCMP that the cluster was being well-used. Subsequent to the installation process, there was a continuous need to coordinate with the HPCMP and the remote sites where the clusters were located. Reports and presentations were required to communicate the appropriate use of this asset.

The decision to split the cluster between two locations; Maui, HI and Dayton, OH; while somewhat cumbersome in use, proved time and again as most propitious, as if one center was down for one reason or the other, the remaining cluster could effectively be used.

Based on the design selected by HPCMP, the JESPP team was able to effectively employ both nodes by using one as a home of the simulation engine and the other as the resident processor for logging and communications. The design decision to provide one site with a 60 gigabyte disk on each node, but leave all of the disks on the other machine in a more easily configured cluster was precipitated by the desire to make the one cluster a "swing machine," allowing easy conversion from UCLASSIFIED to SECRET operations. We determined that this was possible, but problematic. Stability and performance was impacted due to heavy "paging" onto the disk drives on the mesh. Several workarounds were conceived and tested. The desire of JFCOM to have the most stable platform would subsequently lead to the installation of disks on each node of the second cluster.

Major achievements in this area would have to revolve around the successful implementation and operation of meta-computed assets in a real-time, interactive simulation setting. Papers and other exposition to the community seem to suggest the JESPP program is one of the very few projects to consistently and effectively use interactive high-end computing.

3.1.12 Support communications, security and encryption

Not only is the simulation run by JFCOM distributed, but all of the processing and data management is trans-continentally situated. Clearly, sites from throughout the country must communicate effectively, securely and within organizational rules. JESPP personnel became intimately familiar with many of the communications protocols supported by the HPCMP and were instrumental in obtaining necessary changes when those were vital.

The major new paradigm requiring some modification of the rules was the concept of interactive high-end computing. Virtually all of the communications and security rules promulgated and enforced by the HPCMP on its Defense Research and Engineering Network (DREN) assumed and were designed to support batch computing. Interactive computing featuring tens of users and hundreds of independently spawned code processes did not fit well into this milieu.

Supporting on-going communications and encryption efforts as they relate to high performance computing required careful attention. These aspects became more critical as the use of distributed high performance computers was coupled with USJFCOM's desire to include remote sites in the experiments being conducted. Trained and experienced personnel were provided to ensure that this aspect enhanced, rather than inhibited, the achievement of USJFCOM goals.

Most dramatically, the JESPP personnel sought relief from the requirement that all users and processes had to have secure log-in and Kerberized communications. This clearly would have rendered operation impossible and the very restricted nature of the communications, *i.e.* all sites were government facilities that were operated as if the were classified, even if operating at an unclassified level. After significant review and consideration by HPCMP and the subject matter experts, a modified procedure was established and operations were initiated. A notional diagram shows the supported connectivity of virtually all of the runs.

Figure 9 - The Trans-Continental Connectivity, in the Original Tree Configuration

During the effort, the two centers effectively supported a number of events at JFCOM and were able to demonstrate the utility of distributed, interactive, high-end computing.

4.0 Conclusions

The overall results of this project have met or exceeded the results sought by the Joint Forces Command, Joint Experimentation Directorate. The capabilities of the JSAF code to represent more than one million non-military SAF entities has been shown, the operational stability of the system has allowed confident use by USJFCOM operators, performance is acceptable, security issues have been dealt with and all of the assigned tasks have been accomplished.

The code has been effectively operationalized and the ensuing operations have been successfully supported. SPP computing was implemented, a new Distributed Center was stood up, the project successfully was demonstrated to several government leaders and the science generated was published in numerous conference articles.

It was consistently clear that USJFCOM could achieve their success in representing hundreds of thousands of civilian and military entities only with the scalability afforded by the HPCMP hardware and our expertise. This effort gave both proof of that assertion and gave reassurance to those who feared instability.

Members of the USJFCOM J9 team reported to independent inquisitors that "…the SPPs were the most stable part of this project…" in March of 2004 and that continued to be the fact for the rest of the effort. The experience of the sites and of our personnel worked together to avoid problems and produce results.

While providing the day-to-day operational reliability, our team also made several unique and noteworthy advances in SPP operations; including scalable programming, security, data management, data analysis, and visualization. Evidence of this noteworthiness was the fact that several technical papers were presented at conferences and JESPP personnel were invited to speak at three other professional conferences. The papers are included in the Appendix.

Further indicia of the value of this effort were the recognition by both the USJFCOM J9 Directorate, Maj Gen Woods, and the Joint Forces Commander, ADM Giambastiani. The latter has now indicated a desire to advance the time schedule for providing this capability to the warfighter.

Future work is needed and planned in the areas of full implementation of mesh routers, better communications routing, fault tolerance, upgrade to the clusters, support for classified operations, distribution of simulation processes across several nodes where necessary, and the optimization of the initiation and control software package.

Test runs on the mesh router continued to show superiority in internode communications, but complete acceptance into the CVS tree and day-to-day utilization is left for future efforts. The current operation makes, and accepts, limitations in scalability that may not prove acceptable in the future. Plans are made and commitments accepted that will allow full implementation in the future.

Communications, as set forth above, is still running in a star configuration, with the incumbent fault tolerance limitations. In the future, it is planned to implement and test the mesh router as a meta-computing, Wide Area Network (WAN) architecture. This is anticipated to provide both fault tolerance and performance gains.

Fault tolerance can further be advanced by distributing the simulation across several compute platforms and, even, several states. Further, fault tolerance on one Linux cluster mesh will be enhanced if individual simulation processes are "dealt" out and allowed to migrate. In general,

the JSAF program is very amenable to simulation processes dropping out and being inserted without a detrimental impact on stability.

Classified operations will present many new problems that will directly impact the JESPP team. Experience has shown that any perturbation of the platform will impact the rest of the system in ways that were not anticipated. On the other hand, SECRET operations may reduce the necessity of Virtual Network communications and obviate the need for Kerberos.

Several of the simulation packages are sufficiently large to require a single node on their own as opposed to the tens of thousands that may run under different circumstances. These single node programs may benefit from being decomposed over several nodes.

Appendix A – Publications

List of Papers

- Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao, "The Road to Successful Joint Experimentation Starts at the Data Collection Trail", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2003.
- Bill Helfinstine, Mark Torpey & Gene Wagenbreth, "Experimental Interest Management Architecture for DCEE", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2003.
- Robert F. Lucas & Dan M. Davis, "Joint Experimentation on Scalable Parallel Processors", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2003.
- Richard Williams & John J. Tran, "Supporting Distributed Simulation on Scalable Parallel Processor Systems", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2003.
- Dan M. Davis, Garth D. Baer & Thomas D. Gottschalk, "21st Century Simulation: Exploiting High Performance Computing and Data Analysis", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2004.
- Brian Barrett & Thomas D. Gottschalk, "Advanced Message Routing for Scalable Distributed Simulations", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2004.
- Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao, "Successful Joint Experimentation Starts at the Data Collection Trail - Part II", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2004.
- John J. Tran, Jacqueline M. Curiel & Ke-Thia Yao, "An Interdisciplinary Approach to the Study of Battlefield Simulation Systems", Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2004.

Appendix B

The Road to Successful Joint Experimentation Starts at the Data Collection Trail

Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao
M&S Team, Experimentation Engineering Department, J9 USJFCOM
Suffolk, Virginia
rgraeben@ida.org, grafuse@alionscience.com, rmiller@alionscience.com & kyao@isi.edu

ABSTRACT

Joint Forces Command has made great strides formulating the roadmap for conducting joint experiments. However, success for the Command will be measured by its ability to present quantifiable results to support transformational findings. The Services have considerable experience documenting requirements and articulating needs based on quantifiable results. Weapons systems, sensors and related procurement developments lend themselves to statistical testing (primarily through repetitive constructive simulation runs and live tests). The nature of joint experimentation relies on a discovery-type of approach when dealing with 2015 (or later) weapons, decision support systems and identifying the best methods for their utilization. The strengths in using human in the loop (HITL) immersion within distributed virtual simulations (e.g., Joint Semi-Automated Forces (JSAF)), requires innovative approaches to data collection and analysis. The correct approach will provide creditable and quantifiable results to strengthen the Commander, Joint Forces Command's rationale for transformation within DOD. This paper addresses methods for achieving more creditable and quantifiable data support. The first section provides a short description of the spiral development and data integration processes. The second section describes the flexible data collection toolkit used in the initial verification of entity behaviors and performance and then used to extract and display the data generated from the simulations. Finally, the third section describes a distributed framework for scaling the logger and analysis tools to handle very large data sets--in the terabyte range--for meeting the Joint Forces Command, Joint Experimentation Directorate's need for a Distributed Continuous Experimentation Environment capable of providing quantifiable results.

ABOUT THE AUTHORS

Bob Graebener retired after 25 years in the military in 1997. His last active duty assignment was as the Chief of Modeling and Simulation at USACOM's Joint Training, Analysis and Simulation Center. Mr. Graebener is currently a Research Staff Member and team lead with the Joint Semi-Automated Forces (JSAF) program at IDA. In that role, he has participated in several joint experiments sponsored by USJFCOM. He is currently working towards a doctoral degree in Systems Engineering from GWU.

Gregory Rafuse is a data collection analyst and developer and is currently the lead developer for the data collection toolkit. He is a Software Engineer with Alion Science and Technology. Mr. Rafuse has previously served seven years with the US Army as a Field Artillery Crewman. He also possesses an AAS in Computer Information Systems (CIS) from McLennan Community College and is pursuing a BS in CIS from Strayer University.

Robert Miller is a Senior Software Engineer with Alion Science and Technology. He brings over 11 years of experience to the current effort of designing, coding, and testing software for the Future After Action Review System. He holds a Bachelors Degree in Engineering from The Cooper Union School of Engineering and a Masters Degree in Computer Science from the City University of New York.

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University. For his Ph.D. thesis he implemented a spatial and physical reasoning system that automatically generated grids for novel geometries for computational fluid dynamics simulators.

Joint Forces Command (USJFCOM) has made great strides over the past few years in formulating the roadmap and processes necessary to conduct joint experiments. The roadmap covers initial concept design through presentation of results to the Office of the Secretary of Defense. The Joint Requirement Oversight Council (JROC) is the responsible agent for determining which recommendations are taken for action, and more importantly, which are funded. The Commander, USJFCOM's objective is to "provide actionable recommendations from experimentation results to senior leaders to inform options for future force investments" (USJFCOM, 2003 p.3). USJFCOM's effectiveness in this environment will be measured by its continuing ability to present quantifiable results to support joint transformational findings.

Today the modeling and simulation (M&S) and operations research communities are faced with ever increasing challenges to meet the demand for creditable and quantifiable results. An initiative, currently underway at USJFCOM, is attempting to leap ahead of the demand by the creative integration of processes, commercial off the shelf (COTS) products and scalable parallel processors (SPP). This paper will address a method for providing more quantifiable and accurate data generated within large supercomputers running human in the loop (HITL) virtual simulations and federations of simulations used in support of future joint experimentation.[1,2]

BACKGROUND

Leveraging simulation to support joint experimentation has been the centerpiece strategy for USJFCOM because it provides a capability to rapidly prototype futuristic concepts. The M&S toolkit includes constructive as well as virtual simulations and, as the recently concluded Millennium Challenge 2002 joint experiment has demonstrated, live simulations will be integrated when needed (USJFCOM, 2002). Constructive simulations normally provide faster-than-real-time capabilities and are excellent when one requires statistical results to prove or disprove experimental hypotheses. Virtual simulations offer an environment that allows real people to be immersed within the futuristic environment, an excellent medium for evaluating decision-making processes.

Providing Immersive Synthetic Environments

Developing collaborative planning and decision support systems designed to establish user situational awareness present an additional challenge because of the reliance on human subjects as integral components of the command and control system. Immersion of humans within virtual simulations, such as Joint Semi-Automated Forces (JSAF), requires an integrated data generation and collection approach to achieve quantifiable results. Although HITL experiments offer a great potential for exploring complex issues they are a greater challenge to the data collection and analysis team and analysts charged with providing quantifiable results that will survive JROC scrutiny. Being able to provide usable results is also challenged when "setting the initial conditions" requires millions of entities to generate realistic levels of civilian traffic, before one even begins to assess future sensor or weapons systems operating in the urban environment. Federating simulations compounds the challenge, as interactions between simulations have to be checked for accuracy.[3] When sensor systems and radar systems are added to the mix, one can imagine the large quantity of generated data that must be logged, mined and provided to the analyst as quickly and accurately as possible. The enormity of this task was not lost on

[1] The authors caution the reader not to assume that there is only one solution to this challenge. Frequently a number of tools are required to bring distinct, quantifiable results to the senior decision maker.

[2] Federation. A named set of interacting federates, a common federation object model (FOM), and supporting Runtime Infrastructure (RTI), that are used as a whole to achieve some specific objective (Department of Defense, 1998).

[3] Federate. A member of a High Level Architecture (HLA) Federation. All applications participating in a Federation are called Federates (Department of Defense, 1998).

USJFCOM as decisions made in 2002 and executed in 2003 will be realized when the Joint Urban Operations HITL series of experiments in USJFCOM's Distributed Continuous Experimentation Environment gets underway in 2004.

Analysis vs. Discovery

Before describing the joint experimentation data collection strategy, a short description of the difference between analysis and discovery experimentation is warranted. Analytical data is largely derived from statistical testing, applying controls over independent and dependent variables so as to isolate the cause/effect relationships. Constructive modeling has been a major "toolset" in the analysis arena, primarily through its ability to run faster-than-real-time (therefore providing multiple runs to support statistical inquiries). Analysis techniques have been very effective when the problem can be framed with an expected outcome.

Discovery-type events, more often than not, rely on a progressive understanding of what is unfolding, thus requiring flexibility in tool design to explore new avenues as they present themselves. Although hypotheses exist, the number of dependent variables, human interactions, and complexities preclude adherence to rigorous statistical methods of analysis.

THE FUTURE AFTER ACTION REVIEW SYSTEM (FAARS)

The key ingredients to the FAARS are the process, tools and design of logger protocols that operate on scalable parallel processors. The remainder of the paper addresses these three areas. The first section provides a description of the spiral development and data selection and collection processes. The second section describes the flexible data collection toolkit used in the initial verification of entity behaviors and performance and then used to extract and display the data generated from the simulations. Finally, the third section describes a distributed framework for scaling the logger and analysis tools to handle very large data sets--in the terabyte range.

THE PROCESS

Once the concept designers and operations researchers have settled on the experimental concept, collaboration with the M&S community should closely follow. A process for successfully designing the simulation to support specific measures of performance (MOPs) is a recipe for success. Data collection development, when dealing with multiple simulations federated across a wide area network requires a commitment from all functional areas, not just the data collection team. Dependencies are created since the need for specific interactions related to the MOPs must be generated by the entities within the federation or simulation to answer the operations research questions. Figure 1 provides a pictorial of what is involved in the spiral development process (in this case for software).

.

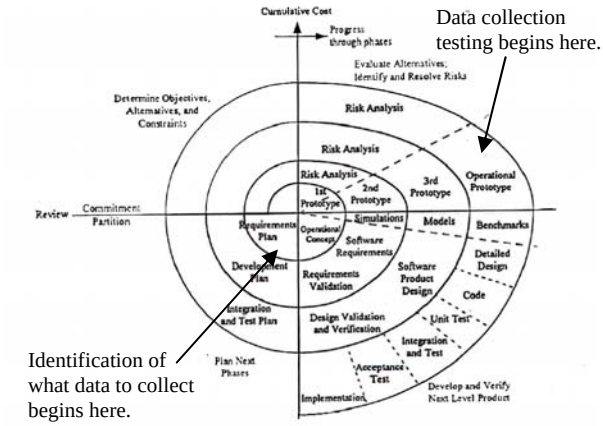


Figure 1. Spiral Development Process (Boehm, 1988)

The following checklist will bring value to the process and establish user confidence in the data and simulations generating the data. The steps described below were designed for a joint experiment supported by the HITL virtual simulation, JSAF (Graebener & Kasputis, 2000).

1. What question(s) does the experiment address?
   -What hypothesis is being explored?
   -Steps include "drilling down" or decomposing the question into sub-elements.

2. What Data will answer the question?
   -Sub-elements are further decomposed into metrics that will support the answer in a manner favorable to quantification.
   -This process begins a spiral crosswalk between the data analyst and the data collection developer to ensure what is being asked for is provided.

3. How is the Data Generated?
   -Further detail generated during the spiral crosswalk identifies how the data is created and by what means the data will be collected.
   -In most cases there will be several methods used to collect the necessary data, while automated data generated from the simulation might be the goal, observers monitoring the test participants could also generate (and collect) data.

4. How is the Data Transmitted/Received?

Table 1 shows the three tests used to explore the question above.[4] Each test has a set of conditions on the first line and then a solution or range of solutions.

Table 1. Data Transmission/Reception Matrix

Test #	Does Entity Transmit?	Does Entity Receive?
1	If answer is: Yes	& if answer is: Yes
	Then: Test to ensure enumerations are accurately sent/received.	
2	If answer is: No	& if answer is: No
	1: Then: Determine if data element can be derived indirectly from other interactions, or: 2: Create a new (experimental) interaction, or: 3: Determine if data element can be generated by non-simulation means?	
3	If answer is: Yes	& if answer is: No
	or	
	If answer is: No	& if answer is: Yes
	1: Then: check to see if the interaction field is empty, and; 2: Fill in the field with appropriate entry, and; 3: Ensure enumerations are accurately sent/received.	

5. When do you need the data and in what format do you need the data products (see Table 2)?

-Identify the data display interval or timings (left-most column).

-The periodicity is negotiable between the data analyst and the data collection developers. In some cases performance issues could impact on how often/how quickly one receives the generated data.

-The format (top row) relates to the following headings:

(1) Raw data.
(2) Grouped.
(3) Combined.
(4) Disparate.
(5) Automatic Visual (Graphs).
(6) Manual Display.
(7) COTS Readable (e.g., SPSS).
(8) Final Form/Report Ready.
(9) Other.

[4] If a gateway is required (e.g., HLA to/from DIS (Distributed Interactive Simulation) further work must be done to ensure an accurate mapping across the gateway is established and tested.

Table 2. Data Format and Timing Checklist

Format	1	2	3	4	5	6	7	8	9
Timing									
Immediate/near-real-time									
Overnight									
End-of-Trial									
End-of-Exp't									
Other									

6. What types of data integration are envisioned for post-experiment processing?

-By Time Segment.
-By Mission/Task.
-By Force Level.
-By Geographic Region/Area.
-Other.

7. What is the cost associated to achieve steps 4 thru 6?

-In processing time?
-In developer time and availability?
-In bandwidth size?
-In relation to other competing interactions?

THE TOOLS

This section describes a framework for using COTS and modified commercial software for logging simulation events and creating analysis tools. The rationale for moving to a new approach in toolkit design is discussed as well as a description of one of the tools currently in use.

New/Innovative Approaches to Data Collection

In previous joint experiment events, the data collection and analysis tools have been custom written applications that collected, processed and performed general analysis activities on a very specific and limited set of data. Previous solutions did not include a level of robustness and reusability necessary to meet future requirements. Because of this, a new approach was created to meet these anticipated demands.

The FAARS toolkit has been designed around a flexible, COTS-based solution minimizing the amount of specific software to be written. The philosophy behind using COTS software over proprietary custom software systems has been to allow concentration of most of the development efforts into providing highly flexible and responsive data extraction methods to support the analysis tools and data displays for the end-users. The FAARS toolkit provides near-real-time event information and a post-event data analysis. The

near-real-time tools supports the verification and validation processes used for quality control of the simulation or federation generated data along with specific, predefined statistical reports and summaries. The post-event tools have been designed to perform analyses in support of the MOPs and measures of effectiveness (MOEs).

FAARS Toolkit Composition

The FAARS toolkit currently is comprised of the following commercial software applications:[5]

Data Collection:
1. hlaResults v2.0.2 (2002)
2. Microsoft Access2000 (1999)

Data Presentation:
1. Near-Real-Time
    a. Apache v1.3.27 (2003)
    b. PHP v4.3.2 (2003)
    c. ChartDirector v3.0 (2003)
2. Post-Event
    a. Microsoft Excel2000 (1999)
    b. Microsoft Visual C++ v6.0 (2002)
    c. MySQL v4.0.11 (2003)
    d. ChartDirector v3.0 (2003)

The core of the data collection is the hlaResults tool. This package provides for the logging of both HLA and DIS-based simulation data. The tool works by intercepting RTI-transmitted information amongst a given federation and storing the collected information in a Microsoft Access2000 database. Figure 2 indicates the flow of data within the FAARS toolkit environment.

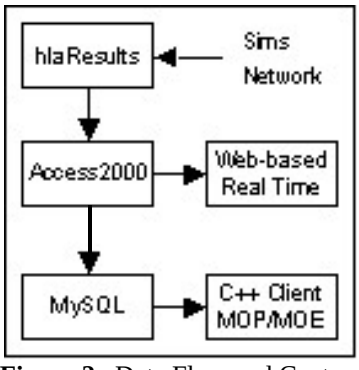


Figure 2. Data Flow and Capture

The near-real-time solution is based on the industry-proven Apache web server, the PHP (Preprocessor Hypertext Preprocessor) scripting language and the Chart Director graphing package. The web server/scripting language combination was primarily chosen to facilitate an independent, cross-browser, operating system solution for providing information in as-close-to-real-time as possible. The web server scripts connect to the hlaResults-generated database using industry standard ODBC (Open Database Connectivity) and SQL (Structured Query Language) commands. The returned information is then processed within the script and presented to an end user via a standard web browser.

The post-event solution is based on a custom-built C++ client interface to access/view data being stored in a relational database (MySQL) using industry standard ODBC and SQL commands. The C++ client interface then processes the returned data and presents the resulting information via a series of either Microsoft Excel2000 spreadsheets, natively or via ChartDirector graphics.

Data Relationships

Storing HLA-generated data in a relational database requires an appreciation for the differences in the way these two technologies handle information. A database, being an information storage and retrieval system, is geared towards eliminating data redundancy. An HLA federation is optimized for data exchange, and therefore is not subject to the limitations of a database. These two goals, though not completely in contradiction, need to be reconciled. Inasmuch as the data is being generated in an HLA-oriented format, it becomes necessary for the database to accommodate the federation (and not the other way around). This "accommodation" essentially amounts to a recognition of the relationships between data attributes in different objects. Such relationships, though not enforced by the HLA federation, must be respected. The FAARS tool development team needed to become highly conversant with the FOM, the Object Model Template (OMT), Federation Execution Document (FED) (a federation agreement between various simulations and the current RTI). From an understanding of how the various parts work together in forming the federation the team determined how the various fields were related and how to "force" relationships between collected data.

To force relationships within data where none are previously defined requires knowledge of the data and the data format. FOM-related data is transmitted as either an interaction between class objects or as a class object status update. Also, there are various linkages used to convey information about a simulation transaction that occurs within the simulation.

[5] COTS application providers are listed in Reference Section.

Understanding hlaResults Collected Data

The hlaResults data collection software package utilizes the OMT, FED and RTI components to generate a "collector." As part of the collector creation process, a pseudo-schema for an Access2000 database is created. This database contains tables representing each interaction or object class as defined by the three components. Each table contains fields representing the "payload" for a specific update of the interaction or class object. When the database is created, the hlaResults-generated schema does not introduce nor define any specific relationships between the various interactions and class objects. Therefore, relationships have to be imposed for the data to be truly relational. It is necessary to decide which fields within the interaction and class object table share common information. Once this is done, a database schema is dynamically generated within the FAARS toolkit and applied to both the Access2000 database (used as the immediate store) and the MySQL database (used as the final "rollup" data store).

Lessons Learned: Evolving Methods and Tools

Flexibility was a going-in design objective for this program. One of the initial decisions was to select Access2000 as the intermediate data store since it was already designed as HLA Results primary database. The team's ability to rapidly prototype the FAARS during early development was facilitated by this decision. However, as the spiral development process matured, new requirements surfaced for increasing the size of collect data storage capability for reasons explained below. The emerging result is a continued improvement to the capabilities of FAARS as it is adaptable to a variety of new COTS database products.

A technical limitation in Access2000 is a 2 GB limit for database sizes. Based on previous usage during test events, a typical HLA federation with 35,000 entities will cause an hlaResults-created Access2000 database to reach this threshold within 2 to 2.5 hrs of event runtime. A typical simulation day, designated by the experiment technical lead, consists of 6 to 8 hrs of continuous runtime, requiring three or more databases to be created. A technique for selecting and processing relevant data from each database (and taking into account overlaps within the data) has been developed by inserting the data into a MySQL database for a "roll-up" into one total event period, to preclude retrieving data separately from each Access2000 database. Currently, two hlaResults collectors are used during experiments in overlapping periods so that there is continuous coverage of simulation data being recorded. Figure 3 shows the method for employing collectors.

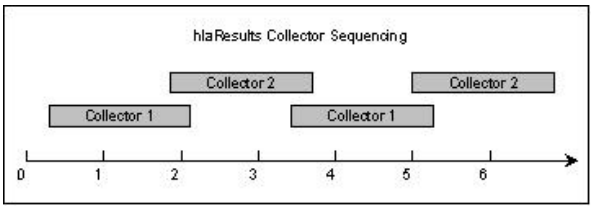


Figure 3. Collector Sequencing over Time

Processing Collected Data

After a simulation run is completed, the collected database segments are processed into the single MySQL database in the following sequence: First, a database schema is applied to the MySQL database that will represent the "rollup" of all of the data collected. Second, the Access2000 database segments are put into sequential order by their unique file names and internal file creation timestamps. Third, each segment is filtered to build a list of relevant tables from which data is to be extracted. Fourth, the first Access2000 database is inserted directly into the MySQL database. Fifth, subsequent Access2000 databases are processed to ignore overlapping data by examining the difference in timestamps between where the previous segment ends and the next segment begins. Also, internal record timestamps are adjusted with an offset so that individual record timestamps represent time since beginning of simulation run and not just for the individual database segment. Finally, summarization information is extracted and stored into new tables to facilitate speed in reviewing common information. Those reports that are processing intensive will be generated and saved for post-event review.

Post-Event Data Processing

Once the simulation data has been processed and inserted into the MySQL database, the MOP/MOE tools are applied to the completed database to provide predefined statistics for the event period. In conjunction with these predefined reports, additional reports and queries can be rapidly created based on additional feedback and desires of the analyst. A sample of predefined reports available for the end user includes: Killer/Victim scoreboard, Entity Lifecycle and Lifecycle Details, and "String" analysis charts. Other MOP/MOE components are developed and included with the toolkit based on input from the data collection and analysis plan designed by the joint experimentation users and analysts.

Aggregate(Top) Level

TARGETS / AGGREGATES	Civ Medium Car	Civ School Bus	Civ Large Truck	Civ Medium Truck	Civ SUV	Civ Small Car	Civ Small Truck	Civ Large Car	Civ Bus	Civ Limo	Maz 543 MEL	UAZ469B	BTR80	Total
High Altitude[1]	234	463	389	240	237	266	230	254	266	218	121	4	3	2925
Medium Altitude[1]	4	12	4	7	6	7	8	4	4	5	3	0	0	64
Totals	238	475	393	247	243	273	238	258	270	223	124	4	3	2989

KEYS	
1.	User-defined aggregation of sensor platforms.

Figure 4. Sensor/Target Scoreboard Example

A Near-Real-Time Tool: Sensor/Target Scoreboard

An example of one of the tools used for near-real-time analysis is the Sensor/Target Scoreboard. The scoreboard is designed to provide insight into the category of class object types being detected, the specific sensor platforms and sensor modes. One possible use of the Sensor/Target scoreboard is to provide summary reports of sensor activity for use during typical daily event briefings.

The scoreboard is divided into four layers of detail, with aggregated sensor platforms versus class object types summaries at the upper-most level. Figure 4 shows a sample Sensor/Target Scoreboard at the aggregate level. The next level of detail (accessed by clicking on a value in the table) displays the total count of individual class objects of a specific type detected by individual platforms and by specific sensor modes. Subsequently, the next level of detail (by clicking on a value in the displayed table) provides a list of specific detected class objects as indicated by the selected sensor platform and sensor mode combination from the previous table. The final level shows the finite information concerning a specific detected class object. The information contained in this report includes details of the sensing platform, the time of the detection(s), the class object detected, location of the detected class object, velocity of the class object (if moving), and detected appearance bit mask value.

SPP & LOGGING DATA

Harnessing SPP for Logging and Analysis

This section describes a distributed framework for scaling the logging and analysis tools to handle very large data sets--in the terabyte range--for meeting J9 needs for the Distributed Continuous Experimentation Environment. This tool is part of USC ISI's Joint Experimentation on Scalable Parallel Processors (JESPP) project. One of the goals of JESPP is to enable modeling of futuristic sensor capabilities, weapon systems and complex battlefield environments by providing the underlying computational and network infrastructure.

Logging and analysis desired properties

Listed below are some of the desired properties for the logger and analyzer framework. Several are inherently contradictory, so design decisions have to be made to balance them.

Scalability
The framework must be able to log and to process very large volumes of data. Based on past experiences from a Future Combat Systems (FCS) experiment, a partial logging of simulation events for selective data analysis requires about 5 kB/hr of data for each entity. Scaling up to the desired million entity SPP runs would require handling ~5 GB/hr. Logging all the simulation events for full playback or comprehensive data analysis would require over 200 kB/hr of data for each entity. For 1 million entities, the data rate is ~200 GB/hr. Five terabytes of data would be collected over the course of a 36 hr experiment.

Minimize resource contention
The simulation and the logger are running simultaneously, both requiring computing resources in terms of CPU, memory, disk access and network bandwidth. The logger should avoid competing against the simulations for resources.

Extensibility
The framework must facilitate the adding of new functional capabilities to handle novel types of analysis. As previously stated, goals and evaluation criteria should be defined well in advance of the event. However, it is not always the case and unanticipated questions do frequently arise. The framework must be

flexible enough to incorporate additional functionality without extensive modification.

Responsiveness
The framework must be able to respond to analysis queries of interest within a prescribed time frame.

Reusability
The framework must be able to work with different simulations and possibly with different HLA runtime infrastructure implementations. Initially, this work focuses on using JSAF and RTI-s (Calvin, et al., 1997).

Partitioning the data

One major design decision for a logger implementation is to determine where to store the logged data. Previous logger implementations for HLA simulations have utilized centralized data storage schemes (e.g., hlaResults) to store events on a centralized relational database. Typically, a centralized logger would self-register as an active listener interested in all federation events. The HLA runtime interface will then forward all events generated by each simulation federate to the logger. However, such centralized storage schemes are difficult to scale up as the number of simulation federates and number of entities increase.

Distributed storage schemes can overcome storage bottlenecks. Instead of funneling all the data to a single computer node, the data is partitioned and saved on multiple nodes. Then the question becomes how to partition the data. Possible partitioning schemes include partitioning by event type, by geography, by time, and by communication connection topology.

There are advantages and disadvantages for accessing data from each partitioning scheme depending on the type of analysis being conducted. For example, in order to compute a Killer/Victim scoreboard, the analyzer needs to access all the Damage Assessment Events. If the data were partitioned by event type, then all the Damage Assessment Events would be on the same node. The analyzer only needs to perform local queries to compute the Killer/Victim scoreboard. However, if analysis involves examining interactions among multiple types of entities, then the analyzer may need to join multiple remote data stores. In this latter scenario partition by geography may be a better choice.

If the desired analysis type is known, the logger should choose the data-partitioning scheme that minimizes remote data access. These partitioning schemes can be viewed as ways of pre-fetching data to support faster computation during analysis. Extra network bandwidth is utilized to aggregate data during collection, in order to improve responsiveness by minimizing communication during analysis. See Figure 5 for three representative types of data aggregation schemes.

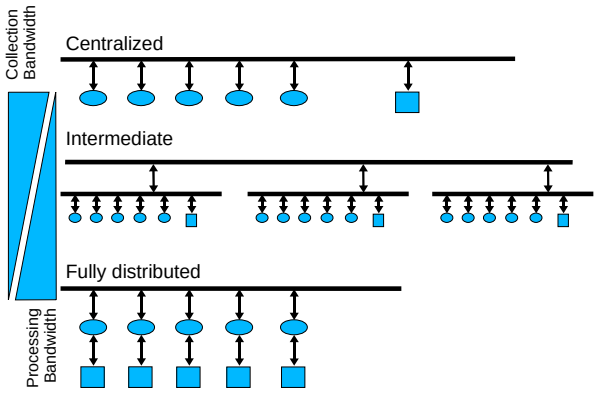


Figure 5. Data partitioning schemes based on communication connection topology.[6]

Frequently the analysis type is not known beforehand and arbitrarily choosing a partitioning scheme wastes valuable bandwidth during the actual simulation run. Also, sometimes multiple analysis types are desired and implementing multiple partitioning schemes wastes even more bandwidth. Resource contention between logger and simulation is unavoidable, when possible the logger should delay resource intensive operations until after the simulation runs. In addition overly aggressive data aggregation may result in underutilization of SPP computing resources. In the extreme centralized scheme only the CPU resource from one node is available for processing.

One way to minimize imposing additional communication load during collection is to partition the data according to the communication connection topology. For example, one type of connection topology used for the SPP is the three-level tree. The simulation federates sit on the leaves of the tree, while the RTI router nodes sit on the internal nodes of the tree. The simulation federates can only communicate with each other through the routers. The logger can be deployed on the routers to passively watch and

[6] Ovals are simulations and squares are the loggers. The centralized scheme expends network resources during collection, but it requires no network communication during processing. The fully distributed scheme is exactly the opposite, which requires no network resource during collection, but expends network resources during processing. Of the three schemes, the fully distributed scheme has the best potential for utilizing the distributed computing resources of the SPP.

collection events as they move through the router. The advantage of passive loggers is that they impose no additional load on the network, since they do not subscribe to interest channels (which forces additional events to be sent). However, the passive loggers attached to routers will fail to capture all the events on RTI implementations (e.g., RTI-s) that perform source-side squelching. If no simulation federate declares an interest to the events then source-side squelching RTI will drop the simulation events before they are sent to the router. To capture all possible events passive loggers attach themselves to the simulation federates themselves before the RTI has a chance to drop the events. Otherwise, the loggers have to become active and explicitly register interest to the squelched events.

Implementation

Based on discussions with potential users and simulation developers, the team came up with three implementation requirements:

- The logger must capture all simulation events. HITL events are very difficult to repeat and the users were concerned that passive data collection would leave gaps.
- The analyzer must be extensible. The user is sometime vague about final analysis products/results, so the analyzer must be flexible enough to handle a variety of analysis types.
- The logger/analyzer must minimize resource contention with the simulation. The simulation developers expressed a strong reluctance to give up simulation performance.

For the initial iteration the team plans to use a fully distributed data storage implementation. Also, to further minimize resource contention the plan is to insert very lightweight probes into the simulations to gather the needed events. Under an HLA simulation environment, federates communicate with each other using the RTI. Conceptually one can place a wrapper around the RTI to intercept all events sent through the RTI. Before forwarding these events to the RTI layer, the wrapper uses non-blocking interprocess communication (IPC) to send the contents of these to separate local logger processors. Since the local logger processor resides on the same node, no network bandwidth is consumed (see Figure 6).

The existence of the wrapper is hidden from the federates, so no modification to simulation federates is needed. Many current RTI implementations, such as RTI-s, RTI-NG and MÄK, provide Interceptors that facilitate wrapper implementation. In addition RTI-s provide dynamic Interceptor loading capabilities, so even recompilations are not needed.

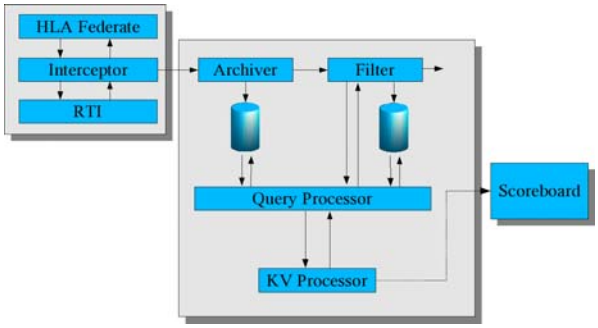


Figure 6. Node level view of logger

The local logger process provides a framework for multiple components to register and receive the intercepted events. For example, note the Archiver and Filter components in Figure 6. The Archiver component faithfully stores all intercepted events to disk. The Filter can be programmed to gather event subsets. Both the Archiver and Filter data stores are accessed through the Query Processor, which exposes an SQL-like API. The actual format of the data stores is exposed. However, for the initial implementation the plan is to store the data in text format. Based on past experiences with logging for network packet analysis, the team has found text format to provide a good first order implementation. The logged data is easy to perform a sanity check on through visual inspection, and easily manipulated with regular Unix tools. The query performance in most cases is acceptable.

For the local data stores the team has also considered using full-fledged relational databases, which provide the flexibility to handle general SQL queries. However, for the initial prototype the decision was not to use relational databases due to concerns that they may not be able to keep up with the volume of data during insertion. Also, logistically it is difficult to set up hundreds of databases (one for each node for each simulation run). In most cases nodes on cluster machines are reserved for fixed time periods. After the time periods expire the local disks are typically wiped clean.

Each Query Processor only provides a partial view of the simulation, since it can only access the data from the local data store. The SPP analyzer provides a framework for integrating multiple local stores to offer the appearance of a centralized data store to application analysis programs (e.g., Killer/Victim scoreboard, etc.).

The scoreboard application sends a SQL query to the root Aggregator (See Figure 7). This query is replicated and forwarded to lower level Aggregators until it reaches the Query Processors at the leaf nodes. Then, the Aggregators merge the results of the query, and send the tables back up the tree. For the initial implementation, the plan is to implement query replication and simple result table merging/sorting operations within the Aggregators. In the future, there are plans to add metadata descriptions to local data stores to enable more advanced processing within the Aggregators.

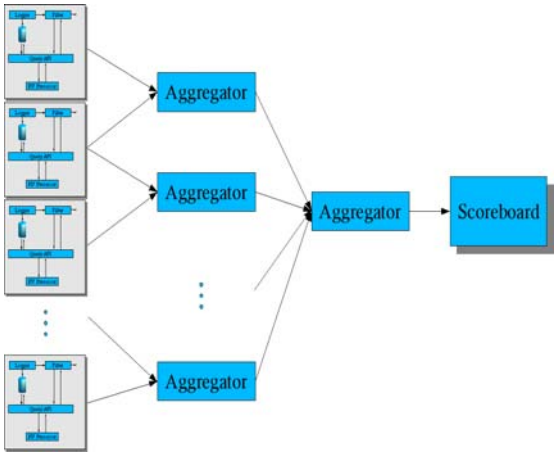


Figure 7. System level view of the logger across the nodes.

CONCLUSIONS

Being able to support senior decision makers with quantitative results does not have to be an insurmountable obstacle when using HITL simulations such as JSAF. Experience being gained at USJFCOM in harnessing scalable parallel processors to do data logging across the M&S federation will provide the technological base to better support the future needs of joint experimentation. Spiral processes and data collection tools round out the knowledge set that will enable quantitative data to be verified, validated and employed in the Department of Defense's quest to transform the military. This exciting area of simulation work is just a segment of the overall support found in joint experimentation, but a critical player, as the prudent application of public funds needs to be as effective as possible.

ACKNOWLEDGEMENTS

The authors wish to thank the JSAF M&S and SPP team members involved in this innovative undertaking. Data collection and analysis is a critical piece of the puzzle, but the people are the key to creative advancement.

REFERENCES

Apache HTTP Server. vers.1.3.27 [Computer Software]. Retrieved May 29, 2002 from http://www.apache.org.

Boehm, Barry W. (1988). A Spiral Model of Software Development & Enhancement. *Computer 21*(5), 61-72.

Calvin, J., Chiang, C., McGarry, S., Rak, S., Van Hook, D., & Salisbury, M. (1997). Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s), *Simulation Interoperability Workshop*, 97S-SIW-019, Mar 1997.

ChartDirector. vers.3.0. [Computer Software]. Retrieved May 12, 2002 from http://www.advsofteng.com.

Department of Defense. (1998). *DOD Modeling and Simulation Glossary*, 1998 [Manual 5000.59-M]. Retrieved November 6, 2002 from https://www.dmso.mil/public/dod/policy.

Graebener, R.J., Kasputis, S. (2000). *The Evolving Role of Analysis in Complex Decision-Making*, Institute for Defense Analyses Document D-2415, (2000).

HlaResults. vers.2.0.2. [Computer Software]. Retrieved November, 18, 2002 from http://www.virtc.com.

MySQL. vers.4.0.11. [Computer Software]. Retrieved May 12, 2002 from http://www.mysql.com.

PHP. vers.4.3.2. [Computer Software]. Retrieved June 10, 2002 from http://www.php.net.

USJFCOM (2002). *Millennium Challenge 02 & the Joint Experimentation Federation*. October, 2003.

USJFCOM (2003). *Campaign Plan 2003-2009 Information Briefing*. March 13, 2003.

Appendix C

Experimental Interest Management Architecture for DCEE

Bill Helfinstine, Mark Torpey
Lockheed Martin Information Systems
Burlington, MA
Bill.Helfinstine@lmco.com, Mark.Torpey@lmco.com

Gene Wagenbreth
Information Sciences Institute
Marina Del Rey, CA
genew@isi.edu

ABSTRACT

The Distributed Continuous Experimentation Environment (DCEE) is a permanent simulation system and facility that is being designed and assembled by the US Joint Forces Command (USJFCOM) to provide a capability to do simulation-backed experimentation without incurring heavy integration and ramp-up costs. Among the several thrusts of the DCEE system is the capability to do large-scale human-in-the-loop experiments in the spirit of the Millennium Challenge 2002 experiment, as well as very detailed representations of joint urban operations scenarios. Additionally, the DCEE system will be used in support of a number of smaller-scale experiments and training events, such as Limited Objective and Multinational Experiments.

In order to provide a system that can scale to a richer and more expansive world, we need to increase the computational power available to produce the environment. However, this leads to a classical problem of parallel computation, where the communications requirements of the system become the bottleneck, and additional computation adds no additional capacity to the system.

This paper describes the architecture that we have prototyped to address some of the problems of data communications scalability. It discusses the interest management techniques that have been used in the past, and how those experiences influenced the prototype design. It talks about the technology that provides finer resolution interest management than simulations have had in the past while allowing better scalability. It explains the limitations of the prototype system and discusses some possible approaches to addressing them. Finally, it describes some likely future requirements of the DCEE system, and talks about how the architecture would have to change in response.

ABOUT THE AUTHORS

BILL HELFINSTINE is a federation developer for the USJFCOM J9 Experiment Engineering Department and a developer and integrator of JSAF (Joint Semi-Automated Forces), as well as primary maintainer and developer of the RTI-s experimental RTI. He has worked in M&S for 9 years, with the last several in support of JFCOM-sponsored exercises, culminating in Millennium Challenge 2002. He is a Staff Software Engineer at Lockheed Martin Information Systems Advanced Simulation Center (LMIS-ASC) in Burlington MA. He received his B.S. in Computer Science and Engineering at the Massachusetts Institute of Technology.

MARK TORPEY is a federation developer for the USJFCOM J9 Experiment Engineering Department and the lead developer and integrator of JSAF (Joint Semi-Automated Forces). His 8 years of M&S experience have been largely in support of JFCOM-sponsored exercises including Millennium Challenge 2002, Unified Vision 2001, and Attack Operations 2000, as well as the DARPA STOW program. He is a Staff Software Engineer at Lockheed Martin Information Systems Advanced Simulation Center (LMIS-ASC) in Burlington MA. He received his M.S. and B.S in Computer Science at the University of Massachusetts.

GENE WAGENBRETH is a parallel processing systems analyst with Information Sciences Institute in Marina Del Rey CA. He has 30 years experience with a range of applications on parallel processors and supercomputers. He received his B.S. in Mathematics and Computer Science at the University of Illinois at Urbana-Champaign.

THE DCEE

The Distributed Continuous Experimentation Environment (DCEE) is a facility and a capability being designed and assembled by the US Joint Forces Command (USJFCOM) for exploration of concepts in joint warfighting. (Ceranowicz et al 2003) One major component of the DCEE is a permanent simulation installation to provide the capability to do simulation-backed experimentation without incurring the major integration and ramp-up costs that previous experiments have incurred.

The simulation capability that will be provided by the DCEE is based on the Joint Experimentation Federation (JEF) that was assembled for the Millennium Challenge 2002 (MC02) experiment. (Ceranowicz et al 2002) This federation provides a framework for the individual services to bring simulation capabilities to a joint virtual world. It provides concept developers the ability to experiment with large-scale battles and situations in a platform-level human-in-the-loop style.

However, the DCEE is not simply designed to be a snapshot of the MC02 version of the JEF; it is designed to evolve and expand to encompass new capabilities and fulfill new requirements as they arise. Therefore, we must keep pushing the technology in advance of the requirements, or the DCEE won't be used. The whole point of the DCEE simulation system is to make it easy and quick to set up a situation and simulate it, in a brainstorming style, in order to bring Joint Experimentation to its full potential.

The simulation component of the DCEE is implemented as a High Level Architecture (HLA) federation. (Dahmann et al 1997) It is an aggregation of a number of simulation systems, each of which has a particular focus on a different facet of the battlefield. However, since many of the simulations that make up the DCEE were originally designed to interoperate using the DIS protocols, (IEEE 1998) they were designed to support scenarios that are in the size range that is supported by DIS—which typically has an upper bound on the number of simulated platforms in the low thousands. This leads to another issue for the DCEE, that of providing a simulation capability that can handle the progressively larger scenarios that the DCEE is designed to handle.

SCALABILITY

One major thrust in the world of Joint Experimentation is that of scalability. The simulated world of MC02, while larger than any others created previously, is not big enough or detailed enough to play out the situations that JFCOM wants to examine. The DCEE must be able to provide a larger, more detailed environment, both in terms of numbers of simulated actors, and the simulated natural environment they interact within.

Table 1. Scalability Achievements Over Time

Event	Object Count	Max Objs Produced PerFederate	Max Objs Consumed PerFederate
STOW97	7000	400	500
J9901	40,000	5000	5000
AO00	160,000	20,000	50,000
MC02	50,000	30,000	30,000
SPP	1,500,000	15,000	70,000

As is shown in Table 1, the number of simulated objects that make up large federations has been steadily increasing, with the exception of MC02, in which it was more important to integrate a large number of new models. The trend towards larger numbers of objects will continue as we move forward, simply because the simulations are not yet capable of portraying a full-scale situation at full accuracy. With the increasing capabilities of computers and networks, we believe that we will be able to produce such a full-scale scenario, but there are still a large number of open issues remaining, both in how to properly control such a simulation, and in how to usefully use and observe such a simulation.

There are a number of factors that limit the size of the simulation that we are able to produce in a system like the DCEE. The computation cost of simulating the objects that make up the world and the interactions between them is the most straightforward of these factors. The communications overhead of sharing these objects between the various simulators introduces another major cost. The final and most complex factor is the effect of the objects simulated by other systems on the local simulated objects.

The computational factor is a straightforward problem to solve, simply by adding additional hardware to the system. However, this solution is constrained due to space restrictions and financial restrictions, and therefore the system is limited in how large and how quickly it can grow.

The communications factor is also constrained by the amount of funding available to support the networking hardware and services to run an exercise. Additionally, there is a significant lead time requirement in order to get networks provisioned and security requirements fulfilled.

The cost of incoming data is still a major subject of research. Each remote vehicle that is received by a simulator adds load to that system. One technology that is often used to reduce the load on a system is interest management. This lets each simulator describe what data might affect its simulation, and the networking infrastructure filters the data that the system needs to consider. While interest management helps an enormous amount, there are cases where it fails simply due to the amount of data requested. (Brunett & Gottschalk 1997)

SCALABLE PARALLEL PROCESSORS

In searching for a possible solution to the first two pieces of the scalability dilemma, we turned to another area of research that has been exploring the areas of scalability and parallelism. The scientific supercomputing community has been exploring the limits of scalability for many years. Furthermore, this community has led to the creation of government-owned and operated High Performance Computing (HPC) centers, many of which are available for use with little lead time.

The HPC centers provide a variety of types of systems, most of which fall into the general category known as Scalable Parallel Processors (SPPs). These systems are defined by their large number of individual CPUs that are connected by a high speed network.

One of the major types of systems run by the HPC centers is the Beowulf cluster. (Sterling et al 1995) Beowulf clusters have become popular systems in the world of HPC systems because of their low cost for the amount of power they provide. A typical configuration of a Beowulf cluster is several hundred commodity PCs running Linux connected with a multi-gigabit network, with custom resource allocation and parallel machine software running on them. Since these clusters are similar to the systems the DCEE uses, we decided to concentrate on using these systems to provide a huge amount of computation and communication resources, and thereby address the time, money, and space restrictions on scalability in the DCEE.

INTEREST MANAGEMENT

The third piece of the scalability question is how to handle the large quantities of data that we can now generate using the capabilities of the SPP systems. We have spent quite a bit of time optimizing the simulation systems to reduce the load imposed by incoming data, but there is an inherent polynomial factor in all simulation systems, simply because vehicles interact with nearby other vehicles, and therefore as vehicle density increases, processing per vehicle increases as well.

In particular, sensor processing is typically an $O(n^2)$ operation on vehicles in a local area. There have been several attempts to mitigate this load through alternative sensor approaches (McGarry & Torpey 1999) (Lorenzo et al 2000) (Kwak & Andrew 2002) but these approaches require additional simulation changes to support them. Due to the legacy nature of many of the DCEE simulation models, these approaches are difficult to implement, since they require modifications to all the different models that are used. This also limits the flexibility of the system, since new simulations have additional requirements over their existing capabilities in order to interoperate with the rest of DCEE.

So, it is still necessary to reduce the quantity of data coming into each simulation to the minimum that they need in order to operate correctly. In general, only the individual simulator can determine what data is interesting, and only at runtime, since the information needed is based on the situation that the simulator is modeling. Therefore, interest management is a

dynamic problem, and needs to react and adapt to changing data requirements by all the components of the system.

This dynamic interest specification and handling is performed by the Data Distribution Management (DDM) functionality of the HLA. The HLA Run Time Infrastructure (RTI) software provides an API to the simulations that allows them to specify what data is interesting in a dynamic fashion. The HLA provides a generalized, abstract way to specify data and interest, by representing data domains as multidimensional spaces, and interest and data specifications as regions within those spaces. (Morse & Steinman 1997) See Figure 1 for a two-dimensional example of how DDM represents interests and data based on overlap of regions.

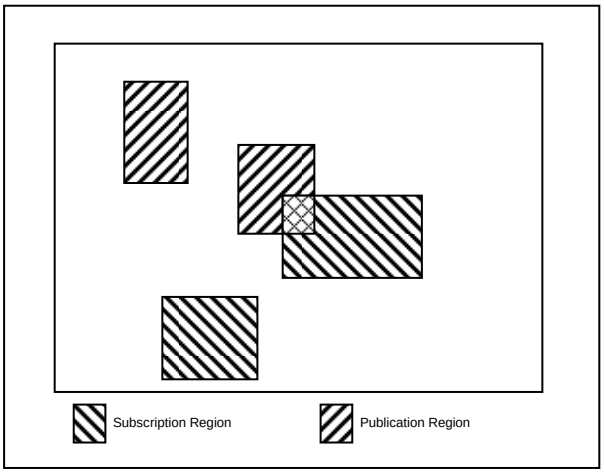


Figure 1. Subscription and Publication Overlap

There have been a number of different DDM designs in several RTI implementations, each of which represents spaces and regions in different ways. The most scalable implementation we have found so far is a statically-assigned grid representation that represents spaces as multidimensional grids, and regions into subsets of the grid. This leads to a fast mapping of interest to grids without any communications and with a simple algorithm. (Helfinstine et al 2001) See Figure 2 for an example of how Figure 1 would be represented in a fixed-gridded implementation.

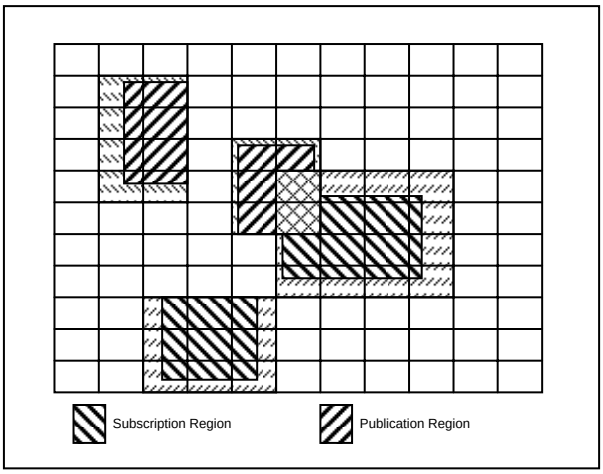


Figure 2. Regions Snapped to a Grid

MULTICASTING

The main communications capability that is provided by the HLA is a publish/subscribe capability that delivers each message to multiple receivers. This capability is often implemented using Internet Protocol (IP) multicast, (Deering 1989) which provides support for point-to-multipoint communications with dynamic subscription changes, over a range of different networking technologies.

However, this presents a problem when trying to run on SPPs, which typically do not support IP multicast. Depending on the type of SPP, it may not support IP at all, since there are many ways to interconnect processors that do not look like a traditional network.

However, SPPs do support message-passing communications, either using IP, as Beowulf systems do, or with some other technology. In order to provide a standardized means of doing message-based communications, the HPC community has standardized on the Message Passing Interface (MPI) as a common API for building parallel programs that express their parallelism in terms of messages. (MPI Forum 1995)

So, it became clear that we would need to build a mechanism that would provide the many-to-many semantics of multicasting while using a communications technology that only supports point-to-point. Furthermore, we also need to maintain our existing capability to run the simulation in a Local Area Network (LAN) environment, since user interfaces and other DCEE federates would not be supported by the SPP.

This led us to examine the work that was done for the Synthetic Forces Express (SF Express) program which investigated running the ModSAF simulation on SPP systems in the 1997 timeframe. (Burnett & Gottschalk 1997-2) This work demonstrated a fairly straightforward way to provide an emulation of the capabilities of multicast using router processes running on SPP nodes to arbitrate the communications and do data duplication and forwarding to appropriate receivers. (Burnett & Gottschalk 1997)

Another interesting body of work that uses a similar networking architecture is being pursued by the DARPA Active Networks program. (Dorsch et al 2002) In their system, software router processes perform data routing to the appropriate recipients in a similar fashion to the SF Express router nodes. This project uses direct region matching to do filtering, which is more precise, but less scalable as the number of regions increases.

RTI-S

In order to construct a system that runs on an SPP and supports HLA federations, we needed an RTI implementation that would use router processes to communicate within the federation. The particular implementation that we used to form the basis of this system is the RTI-s subset RTI implementation. (Calvin et al 1997)

We chose this implementation for several reasons. It was available to us with source code, and is familiar to us from its use in previous experiments, so it was easily modifiable to use the new communications system we were building. It has much less code than a full RTI implementation, which makes it much easier to understand and extend. It scales well, and has a fairly small memory footprint. Finally, it has a very flexible implementation of DDM, providing multiple static inset grids that allow detailed tuning of interest specifications. (Rak et al 1997)

COMMUNICATIONS ARCHITECTURE

We put together a design for the communications architecture based on the concept of stackable protocol modules. We analyzed the existing RTI-s communications code and refactored the functionality it provided into several pieces.

The original RTI-s network interface is composed of the stream manager classes, which provide single-sender to multiple-receiver message sending, receiving, and subscription, and the message buffer class, which provides an interface to messages. Below this interface, the infrastructure provides message bundling, to reduce the packet count by aggregating multiple small messages into each packet, and fragmentation, to split large messages into multiple packets and reassemble them on receive. Finally, it sends and receives the actual packets using IP multicast.

Then, these main components of the communications infrastructure were separated out into chained protocol modules, and given a standardized interface to ease extension and flexibility. We then added additional modules that send and receive packets using point-to-point TCP and point-to-point UDP. Finally, we added a module that translated generalized subscription requests into a message that states the current list of subscriptions, which is sent across the point-to-point connection and remembered by the receiver. Figure 3 shows three possible configurations for an RTI communications structure, with the three columns of protocol modules below the stream manager.

In order to operate on SPP systems that use MPI as their connectivity basis, we built an MPI send and receive module. However, we were worried about the fault-tolerance effects of MPI, and since we were running on Beowulf clusters, which support IP connectivity, we ended up using TCP for our prototype events.

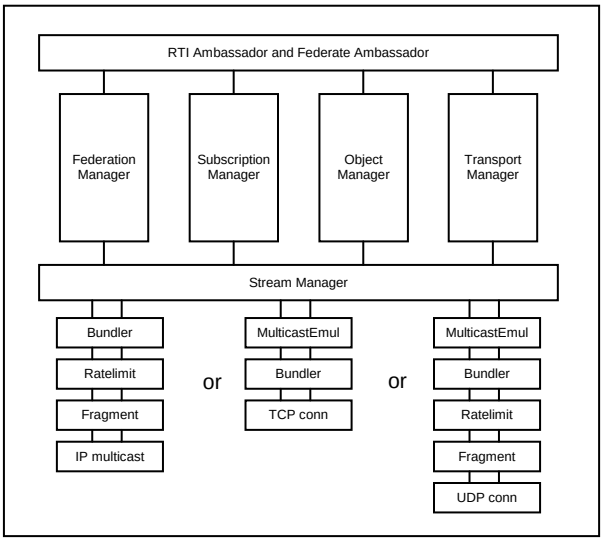


Figure 3. Three Example Configurations of the RTI-s Communications Infrastructure

This general model of protocol modules has allowed us to experiment with additional message transformations depending on our needs. In this vein, we built a module that compresses the data across a connection, when our bandwidth is low and we have available CPU time. For testing purposes, we also built a module that simulates a lossy network, and which randomly drops incoming or outgoing data with a specified loss rate. We see this as a very convenient way of integrating future data transformations as they become necessary.

ROUTER DESIGN

Once we had a way for the RTI to send and receive data in a point-to-point fashion, we needed a router implementation that would receive the data from each federate and forward it to the clients that subscribed to it. As an initial implementation, we built a simple router process that reuses the RTI's flexible connection code, receiving the data and processing it in the same fashion as the RTI. Figure 4 provides a diagram of a router that is routing between three connections.

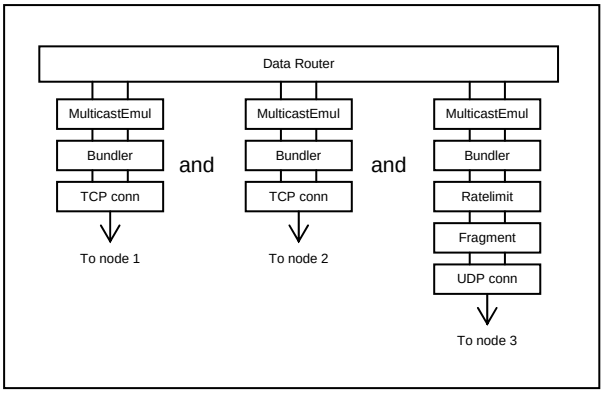


Figure 4. Simple Router Design Routing Between Connections to Three Federates or Other Routers

In order to send messages to only those receivers that want to receive it, each connection tracks what the receiver's subscriptions are. Therefore, since each connection has knowledge of what the receiver wants to hear, it can filter outgoing data before it makes it through the protocol chain. Since each side of each connection knows this information, if no listeners in the system want to hear a particular piece of data, it won't be sent out of the originating machine. This aggressive source-side squelching of data is a very nice side-effect of the router design.

However, in order to accomplish this, we need to send subscription information across each link in both directions. In the case of the router, it turns out to be fairly simple-- a router's interest is the union of all its connections' interests. Therefore, the two major things that a router must do is to forward incoming messages to all other connections, and update interest information on all other connections when one connection changes.

TOPOLOGY

This simple router architecture is quite functional, but it does have some significant problems. In particular, it does not handle cycles in the graph of routers. Each router expects to be able to forward all incoming messages to all receivers. If one of those receivers is a router that forwards a message to a router that has already forwarded it once, the routers will send data in a loop forever and overload the system. However, this implies that these routers can only be set up in a tree structure if we have more load than a single router can handle. This is obviously not a scalable design.

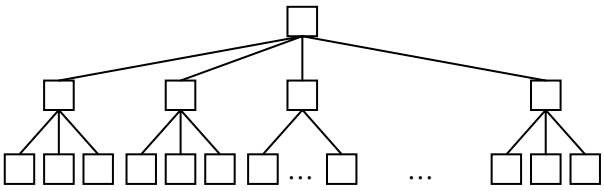


Figure 5. Simple Tree-Based Router Topology

Because of this, we also built a second router implementation based on the up/down fully-connected mesh topology that was explored by the SF Express project. Unfortunately due to schedule pressure, we have not yet been able to test this design fully.

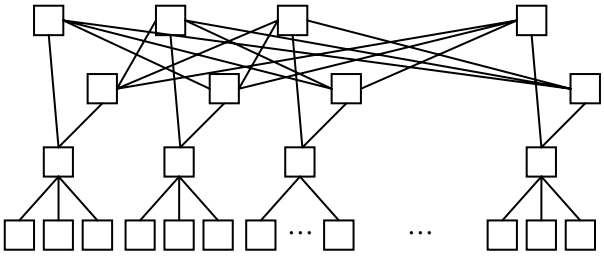


Figure 6. Triplet-Connected Mesh Topology

We believe that we need to investigate the topic of topology more, and look into new ways to organize the communications between the various components of the federation. In particular, when Wide Area Network (WAN) connections between multiple SPP systems are introduced, being constrained to a tree structure can result in very heavy data loads to one of the sites,

which is a very expensive solution to a software limitation. Further, WAN connections have a much lower bandwidth than SPP interconnects or LAN connections, and therefore it makes sense to investigate specialized connection methods across WANs, and different tradeoffs in the design of the communications setup.

INTEREST MANAGEMENT IMPROVEMENTS

One of the results of this new messaging architecture was that we began to run into limitations of the RTI-s interest management infrastructure. In particular, we wanted to expand the number of interest states from the previous maximum of 3000 to 100000 or more. Previously we were limited by the capabilities of IP multicast routers, which begin to fail after roughly 3000 multicast groups, but with our own router implementation, we no longer are subject to these limits. Since the efficacy of the static grid interest management scheme is determined by the number of interest states, the more states that are available, the smaller the grid cells are. As the grid cells become smaller, less unwanted data will be delivered to the federate.

However, the existing code began to perform poorly, due to the use of arrays of integers to represent the list of interests of a particular subscription. In order to scale the number of interest states up, we had to refit a number of internal data types in RTI-s to be more efficient, both in storage usage and in access time. In particular, the list of interest states was changed to be represented as a sparse bit vector implementation, with a fixed-block-size representation. This provided a way to quickly determine interest overlap as well as a fast means of calculating the union of interests in the router. Further, it resulted in a compact representation that could easily be sent over connections with a fairly small overhead.

Similar changes were made throughout the RTI code, in many places where the assumption was that an array of values with an entry for each interest state would be acceptable, we had to change to a tree representation or a hash table in order to not consume large amounts of memory. Additional changes were required to provide a means of associating objects with their interests in an efficient fashion.

Finally, a centralized means of recording statistics about data amounts and counts was added, in order to be able to pinpoint pieces of the system and what was causing slowdowns. With the existing RTI-s capability to examine internal information, this allows a remote, distributed debugging capability that was extremely useful in monitoring the system as it ran.

PROTOTYPE EVENTS

We ran two prototype events, in which we demonstrated that it is possible to generate enormous numbers of vehicles in a very large virtual environment, using SPP systems. Both events were run using a subset of the DCEE federation, composed of the JSAF simulation GUIs, the JSAF simulator running aircraft, ships, and ground combatants, and the JSAF clutter simulator providing background and civilian traffic.

In December 2002, we were able to generate over 1,000,000 vehicles, using a terrain that covered the entire Pacific Rim. The simulation ran on the University of Southern California's Beowulf cluster, and operators and observers were located at Joint Forces Command in Suffolk, Virginia, as well as at Information Sciences Institute in Los Angeles. We were able to use 50,000 interest states to provide a fairly precise specification of interests, in several geographically disparate simulated locations.

In March 2003, we ran an even larger event, generating over 1,500,000 vehicles on the same terrain database, but located in different areas with more terrain detail. We ran on the Huinalu Beowulf cluster at the Maui High Performance Computing Center and the ASC Beowulf cluster at Wright-Patterson AFB, with observers in Suffolk and Los Angeles again. We also increased the number of interest states to 100,000 without adverse effect.

Both of these events were focused on testing the functionality of the new system, and showed that we can indeed generate a very large simulated environment. They also demonstrated that we have quite a bit of additional work that we can do, in order to make the system viable for the end users. In particular, WAN latencies and inefficiencies in the simulation's control protocols combine to make the user interfaces very sluggish. The tree nature of the routers also became a point of failure when the system was under its heaviest loads. We suffered a number of router failures due to data overload, and we are still working to address these.

FUTURE REQUIREMENTS

We still face a number of issues that we need to resolve in order to make the use of SPP systems possible for DCEE. The events that we have run so far show major promise, but have not yet demonstrated that we are able to fulfill the DCEE's flexibility and ease-of-use requirements yet.

The first major requirement is that we need to make it much easier for non-experts to acquire time on SPPs and execute the system on them. It currently is a fairly involved process that takes several people to accomplish. This is a major project that is already underway. (Williams & Tran 2003) An initial version of the MARCI launch and control system was tested at the March event, and it is undergoing further development and refinement.

Another major requirement is that we must have a way for simulations running on SPP systems to participate in the DCEE federation. The primary reason we cannot simply plug the SPP systems into the DCEE is that the DCEE uses the enhanced version of RTI-NG developed for Millennium Challenge 2002, (Hyett & Wuerfel 2003) and the SPP uses RTI-s with point-to-point routers. Since they use different RTI implementations, they run in two separate HLA federations, and we need to build a federation gateway that will allow us to bridge data back and forth between the two federations. This is not an easy task (Granowetter 2003) but we believe that we can build such a gateway as long as its scope is restricted to the DCEE and similar federations. This is another ongoing major project.

An additional issue that we are beginning to investigate is how the SPP will help analysts do After Action Review of the huge amounts of data that can be produced by simulations running on an SPP. A distributed logging and query system is currently being designed to attempt to address this requirement.

One of the most important areas that we need to investigate is the issue of control. As we scale up scenarios to the desired sizes, it becomes more and more difficult to control the simulation and make sure it behaves in a proper fashion. We need to look into schemes that reduce the amount of operator control that is required to run a simulation. This would have an additional benefit for DCEE as well, since any technique that reduces the number of personnel involved will be of incredible utility.

CONCLUSIONS

The use of Scalable Parallel Processor systems has a great deal of promise in building larger and more detailed virtual environments, both for experimentation and for many other uses of simulation. We are integrating the use of SPP systems into the DCEE, and we believe that it will provide an extremely valuable asset in the DCEE environment.

The use of software interest management routers to provide data distribution gives us a great deal of flexibility in building a scalable system and providing the building blocks to more detailed dataflow control and management.

There are many additional dimensions that are worth exploring, both in better integration into DCEE, and additional technical exploration to discover new ways to apply the SPP assets to the problems of DCEE and similar human-in-the-loop simulation systems.

ACKNOWLEDGEMENTS

This material is based in part on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Brunett, S. & Gottschalk, T. (1997). An Architecture for Large ModSAF Simulations Using Scalable Parallel Processors, *Center for Advanced Computing Research Technical Report* CACR-155.

Brunett, S. & Gottschalk, T. (1997). Scalable ModSAF Simulations With More Than 50,000 Vehicles Using Multiple Scalable Parallel Processors, *Center for Advanced Computing Research Technical Report* CACR-156.

Ceranowicz, A., Torpey, M., Helfinstine, B., Evans, J., & Hines, J. (2002). Reflections on Building the Joint Experimental Federation, *Proceedings of the 2002 Interservice/Industry Training, Simulation and Education Conference.*

Ceranowicz, A., Dehncke, R., Cerri, T., & Blank, J., (2003). Moving toward a Distributed Continuous Experimentation, *Submitted to the 2003 Interservice/Industry Training, Simulation and Education Conference.*

Calvin, J., Chiang, C., McGarry, S., Rak, S., Van Hook, D., & Salisbury, M. (1997). Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s), *Proceedings of the Spring 1997 Simulation Interoperability Workshop*, Paper 97S-SIW-019.

Dahmann, J., Fujimoto, R., & Weatherly, R. (1997). The Department of Defense High Level Architecture, *Proceedings of the 1997 Winter Simulation Conference*.

Deering, S. (1989). *Host Extensions for IP Multicasting*, IETF Network Working Group, RFC 1112.

Dorsch, M., Kostas, T., & Skowronski, V. (2002). Reducing Bandwidth Requirements of Distributed Simulations, *Proceedings of the Fall 2002 Simulation Interoperability Workshop*, Paper 02F-SIW-118.

Granowetter, L. (2003). RTI Interoperability Issues - API Standards, Wire Standards, and RTI Bridges, *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, Paper 03S-SIW-063.

Helfinstine, B., Wilbert, D., Torpey, M., & Civinskas, W. (2001). Experiences with Data Distribution Management in Large-Scale Federations, *Proceedings of the Fall 2001 Simulation Interoperability Workshop*, Paper 01F-SIW-032.

Hyett, M. & Wuerfel, R. (2003). Connectionless Mode and User Defined DDM in RTI-NG V6, *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, Paper 03S-SIW-102.

IEEE (1998). *IEEE Standard for Distributed Interactive Simulation - Application Protocols*, IEEE Std 1278.1A-1998.

Kwak, D. & Andrew, E. (2002). Technical Challenges for Joint Synthetic Battlespace (JSB), *Proceedings of the Fall 2002 Simulation Interoperability Workshop*, Paper 02F-SIW-075.

Lorenzo, M., Morse, K., Riggs, B., & Rizik, P. (2000). Sensor Simulation Scalability Using Composable Component Federates, *Proceedings of the Fall 2000 Simulation Interoperability Workshop*, Paper 00F-SIW-090.

McGarry, S. & Torpey, M. (1999). Back to Basics: Balancing Computation and Bandwidth, *Proceedings of the Fall 1999 Simulation Interoperability Workshop*, Paper 99F-SIW-188.

Morse, K. & Steinman, J. (1997). Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering, *Proceedings of the Spring 1997 Simulation Interoperability Workshop*, Paper 97S-SIW-052.

MPI Forum (1995). *MPI: A Message Passing Interface Standard*, Version 1.1.

Rak, S., Salisbury, M., MacDonald, R. (1997). HLA/RTI Data Distribution Management in the Synthetic Theater of War, *Proceedings of the Fall 1997 Simulation Interoperability Workshop*, Paper 97F-SIW-119.

Sterling, T., Becker, D., Savarese, D., Dorband, J., Ranawake, U., & Packer, C. (1995). Beowulf: A Parallel Workstation for Scientific Computation, *Proceedings of the 24th International Conference on Parallel Processing*.

Williams, R. & Tran, J. (2003). Supporting Distributed Simulation on Scalable Parallel Processor Systems, *Submitted to the 2003 Interservice/Industry Training, Simulation and Education Conference.*

Appendix D

Joint Experimentation on Scalable Parallel Processors

Robert F. Lucas, Dan M. Davis
Information Sciences Institute, University of Southern California
Marina del Rey, California
rflucas@isi.edu, ddavis@isi.edu

ABSTRACT

The JESPP project exemplifies the ready utility of High Performance computing for large-scale simulations. J9, the Joint Experimentation Program at the US Joint Forces Command, is tasked with ensuring that the United States' armed forces benefit from improvements in doctrine, interoperability, and integration. In order to simulate the future battlespace, J9 must expand the capabilities of its JSAF code along several critical axes: continuous experimentation, number of entities, behaviors complexity, terrain databases, dynamic infrastructure representations, environmental models, and analytical capabilities. Increasing the size and complexity of JSAF exercises in turn requires increasing the computing resources available to JFCOM. Our strategy exploits the scalable parallel processors (SPPs) deployed by DoD's High Performance Computing Modernization Program (HPCMP). Synthetic forces have long run in parallel on inter-networked computers. SPPs are a natural extension of this, providing a large number of processors, inter-connected with a high performance switch, and a collective job management framework. To effectively use an SPP, we developed software routers that replace multicast messaging with point-to-point transmission of interest-managed packets. This in turn required development of a new simulation preparation utility to define the communication topology and initialize the exercise. We also developed tools to monitor processor and network loading and loggers capable of absorbing all of the exercise data. We will report on the results of J9's December 2002 Prototype Event which simulated more than one million clutter entities along with a few thousand operational entities using 50,000 interest states on a terrain database encompassing the entire Pacific Rim. The exercise was controlled and "fought" from a J9 test bay in Suffolk, VA and the clutter entities were executed on a remote SPP in Los Angeles, CA. We will also present results from the Prototype Event in March 2003, as well as our long-term plans.

ABOUT THE AUTHORS

Robert F. Lucas is the Director of the Computational Sciences Division of the University of Southern California's Information Sciences Institute (ISI). There he manages research in computer architecture, VLSI, compilers and other software tools. He has been the principal investigator on the JESPP project since its inception in the Spring of 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory, the Deputy Director of DARPA's Information Technology Office, and a member of the research staff of the Institute for Defense Analysis's Center for Computing Sciences. From 1979 to 1984 he was a member of the Technical Staff of the Hughes Aircraft Company. Dr. Lucas received his BS, MS, and PhD degrees in Electrical Engineering from Stanford University in 1980, 1983, and 1988 respectively.

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California, and has been active in large-scale distributed simulations for the DoD. While he was the Assistant Director of the Center for Advanced Computing Research at the Caltech, he managed Synthetic Forces Express, a multi-year simulation project. Prior to that, he was a Software Engineer on the All Source Analysis System project at the Jet Propulsion Laboratory and worked on a classified project at Martin Marietta, Denver. An active duty Marine Cryptologist, he currently holds a U.S.N.R. commission as a Commander, Cryptologic Specialty. He has served as the Chairman of the Coalition of Academic Supercomputing Centers and the Coalition for Academic Scientific Computation. He was part of the University of Hawai'i team that won the Maui High Performance Computing Center contract in May of 2001. He received a B.A. and a J.D., both from the University of Colorado in Boulder.

Introduction and Background

The United States has a vested interest in being able to simulate more than one million vehicles, all with sophisticated behaviors, operating on a global-scale, variable resolution terrain database. This is driven by the government's needs to accommodate new computer and communications technology (Cebrowski, 1998) and simulate more complex human functions in technically diverse situations (Sanne, 1999). The U.S. Department of Defense (DoD) has begun a series of experiments to model and simulate the complexities of urban environments. In support of their mission, analysts need to conduct interactive experiments with entity-level simulations, using programs such as the Semi-Automated Forces (SAF) family used by the DoD (Ceranowicz, 2002). This needs to be done at a scale and level of resolution adequate for modeling the complexities of military operations in urban situations. All of this mandates the government analysts' requirement of simulations of at least 1,000,000 vehicles or entities on a global-scale terrain database with high-resolution insets. Experimenters using large numbers of Linux PCs distributed across a LAN found that communications limited the analysts to tens of thousands of vehicles, about two orders of magnitude fewer vehicles than their needs. This paper addresses the benefits of the successful application of computational science and parallel computing on SPPs to this issue. By extension, it illuminates the way for those with similar simulation needs, but faced with similar computational constraints, to make beneficial use of the SPP assets of the High Performance Modernization Program (HPCMP.)

While there are many approaches that are currently in use, simulation and modeling at the entity level (modeling each individual person and vehicle) manifest some very attractive features, both for training and for analysis. Many who would argue that entity level simulations should be employed, maintain that these would generate the most timely, most valid, and most cost-effective analyses. Making these simulations so that the DoD can involve humans, i.e. Human-in-the-Loop (HITL), additionally augments the DoD's ability to assess true impacts on personnel and procedures. (Ben-Ari, 1998) There are several new methods to modeling human behavior (Hill, 2000). While these require significant independent research (vanLent, 1998), they also require significant additional compute power. Current capability does not allow the analyst to conduct these experiments at the scale and level of resolution necessary. These constraints have also been found in other varieties of simulation.

In the present case, newfound emphasis on civilian, "White," and clutter entities has expanded the horizons of entity-count by an order of magnitude. Take any urban setting. The number of civilian vehicles will easily outnumber the combat vehicles by a factor of ten, and more likely, by a factor of 100. Trying to assess the utility of sensors in discriminating the former from the latter will be ill served by a simulation that is limited to a few thousand vehicles total.

In order to make good use of the SPP assets currently available to DoD experimenters, the JESPP project applied approaches that others should find easily and reliably implementable on other, similar, efforts. The discussion of the implementation of the JESPP code into the JSAF code base will not only represent a record of where we have been, but show the path for where we may go in the future.

The current work on Joint Experimentation on Scalable Parallel Processor (JESPP) Linux clusters enabled successful simulation of 1,000,000 entities. Software implementations stressing efficient inter-node communications were necessary to achieve the desired scalability. One major advance was the design of two different software routers to efficiently route information to differing hierarchies of simulation nodes. Both the "Tree" and the "Mesh" routers were implemented and tested. Additionally, implementations of both MPI and Socket-Programmed variants were intended to make the application more universally portable and more organizationally palatable. The development of a visual and digital performance tool to monitor the distributed computing assets was also a goal that has been accomplished, leading to insights gained by using the new tool. The design and selection of competing program initiation tools for so large a simulation platform was problematical and the use of existing tools was considered less than optimal. The analytical process for resolving initiation issues, as well as the design and implementation of the resulting initiation tool developed by the group, is both a demonstrable result and the foundation of a computation science paradigm for approaching such problems. The design constraints faced are analyzed

along with a critical look at the relative success at meeting those constraints.

The requirements are for a truly interactive simulation that is scalable along the dimensions of complexity of entity behavior, quantity of total simulated entities, sophistication of environmental effects, resolution of terrain, and dynamism of features. This is a challenge that the authors assert may only be amenable to meta-computing across widely dispersed and heterogeneous parallel computer assets (Foster, 1997). Just achieving scalability in all of these dimensions would be difficult. Even more so, fielding a stable, dynamically reconfigurable compute platform that may include large parallel computers, Linux clusters, PCs on LANs, legacy simulators, and other heterogeneous configurations produces new obstacles to implementation. Several unique and effective Computational Science approaches are identified and explained, along with the possible synergy with other Computational Science areas.

The current work is based on the early work headed by Paul Messina at Caltech (Messina, 1997). The Synthetic Forces Express project (SF Express) began in 1996 to explore the utility of Scalable Parallel Processors (SPPs) as a solution to the communications bottlenecks then being experienced by one of the conventional SAFs, ModSAF. The SF Express charter was to demonstrate a scalable communications architecture simulating 50K vehicles on multiple SPPs: an order-of-magnitude increase over the size of an earlier major simulation.

SPPs provided a much better alternative to networked workstations for large-scale ModSAF runs. Most of the processors on an SPP can be devoted to independent executions of SAFSim, the basic ModSAF simulator code. The reliable high-speed communications fabric between processors on an SPP typically gives better performance than standard switching technology among networked workstations. A scalable communications scheme was conceived, designed and implemented in three main steps:

1. Individual data messages were associated with specific interest class indices, and procedures were developed for evaluating the total interest state of an individual simulation processor.

2. WAN Communications: Within an individual SPP, certain processors were designated as message routers; the number of processors used as routers could be selected for each run. These processors received and stored interest declarations from the simulator nodes and moved simulation data packets according to the interest declarations.

3. Inter-node Communications: Additional interest-restricted data exchange procedures were developed to support SF Express execution across multiple SPPs. The primary technical challenge in porting ModSAF to run efficiently on SPPs lay in constructing a suitable network of message-passing router nodes/processors. SF Express used point-to-point SPP MPI communications to replace the UDP socket calls of standard ModSAF. The network of routers managed SPP message traffic, effecting reliable interest-restricted communications among simulator nodes. This strategy allowed considerable freedom in constructing the router node network.

As the simulation problem size increased beyond the capabilities of any single SPP, additional interest-restricted communications procedures were needed to enable Metacomputed ModSAF runs on multiple SPPs. After a number of options were considered, an implementation using dedicated Gateway processors to manage inter-SPP communications was selected.

In March of 1998, the SF Express project performed a simulation run, with more than 100,000 individually simulated vehicles. The runs used several different types of Scalable Parallel Processors (SPPs) at nine separate sites spanning seven time zones. These sites were linked by a variety of wide-area networks. (Brunett, 1997)

This work depended on the existing DIS standard utilized by the SAFs at that time. That standard was replaced by the HLA/RTI standard that was purportedly more scalable, but several years of use has shown the clear limits of this simulation approach. This has not prevented some experimenters from getting very good results while simulating ~ 30,000 entities (Ceranowicz, 2002). These new standards and additional requirements have driven the development of two new router designs, Mesh Routers and Tree Routers.

JSAF

The Joint SemiAutomated Forces (JSAF) is used by the US Joint Forces command in its experimentation efforts. JSAF runs on a network of processors, which communicate, via a local or wide area network. Communication is implemented with High Level Architecture (HLA) and a custom version of Runtime Infrastructure (RTI) software version RTIS. A run is implemented as a federation of simulators or clients. Multiple clients in addition to JSAF are typically included in a simulation.

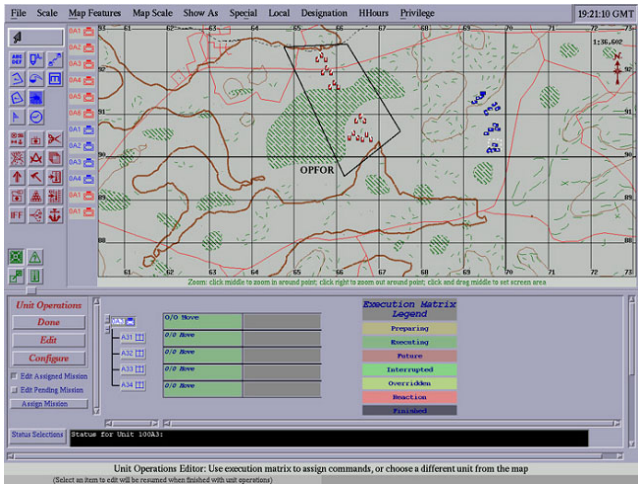


Figure 1
Plan View display from a SAF

HLA and RTI use the publish/subscribe model for communication between processors. Typically, these processors are relatively powerful PCs using the Linux operating system. A data item is associated with an interest set. Each JSAF instance subscribes to ranges of interest. A JSAF may be interested in, for example, a geographic area or a range of radio frequencies. When a data item is published the RTI must send it to all interested clients.

A typical JSAF run simulates a few thousand entities using a few workstations on a local area network (LAN). A simple broadcast of all data to all nodes is sufficient for this size simulation. The RTI on each node discards data that is not of interest to each receiving node. Broadcast is not sufficient when the simulation is extended to tens of thousands of entities and scores of workstations. UDP multicast was implemented to replace the simple broadcast. Each simulator receives only the data to which it has subscribed, i.e. in which it has a stated interest.

Figure 2
3D Rendered display from a SAF

Operational imperatives drive experimental designs that now required further expansions of JSAF capabilities. As noted before, some of the requirements justifying these extensions are the need for:

- More entities
- More complex entities
- Larger geographic area
- Multiple resolution terrain
- More complex environments

The most readily available source of one or more orders of magnitude of increased compute power is the capability presented by Scalable Parallel Processors. In the JESPP project, JSAF was ported to run on multiple Linux clusters, using hundreds of processors on each cluster. Future runs will require thousands of processors on multiple clusters. The primary difficulty in using these resources is the scaling of internode communication.

UDP multicast is limited to approximately three thousand different channels. Based on geography alone, worldwide simulations using JSAF require many more interest states. UDP multicast has been replaced by software routers.

Software routers were implemented on individual nodes in a network that includes all of the client simulators. Each simulator is connected to only one router. Routers are connected to multiple clients and multiple routers. Each connection is a two-way connection. Two types of information are present in the network. One is data along with interest description. The other is the current interest state of each client. The interest state changes as each node subscribes and unsubscribes to specific interest sets, as is appropriate depending on the simulation progress.

Each router must maintain the interest set of each node to which it is connected, including other routers. A router's interest set is the union of all connected nodes. A router then uses the interest state associated with data it receives to determine how to forward the data. For a given topology communication is minimized such that each client node receives exactly the data in which it is interested.

The initial router implementation was a tree router. Each router has multiple clients but only one parent. There is one router that is the top of the tree. A second topology has subsequently been implemented. We have referred to it as a mesh router. Instead of a single router at the top of a tree, there is a mesh of routers with all to all communication. Each simulator is a client of one of the mesh routers. Like the tree router, the primary task of the mesh router is to maintain the interest state of all

clients so as to forward only data that is of interest to each client and router. Further hybrid topologies are possible with little or no code modification, such as a mesh of meshes or a mesh of trees. Conceptually, the mesh should provide better scalability.

Another use of routers is the implementation of gateways providing an interface between different RTI and communication implementations. Both TCP and UDP are used for communication. Routers can use a different protocol on different connections and perform required data bundling, unbundling, etc. Different RTI implementations, required by simulators developed by different groups, can communicate via router-based gateways.

The ultimate goal is for the capacity of a simulator network to scale easily as the number of processors is increased by several orders of magnitude. Comprehensive testing and measurement is required to document the performance of various topologies and router implementations. This testing will identify performance bottlenecks and suggest alternative implementations to be tested. Multiple simulation scenarios must be tested to construct guidelines for assigning simulators, routers and topologies to multiple SPPs.

Fault tolerance is another area being studied. JSAF simulators are not affected by the loss of other simulators. The use of routers creates a single point whose failure eliminates multiple simulators. The use of dynamic topology will be studied and implemented to minimize the consequences of single node failures. Several different concepts of providing redundancy or instantaneous recovery are being considered and will be implemented and evaluated.

Tree Routers

The first router implementation is a tree router.

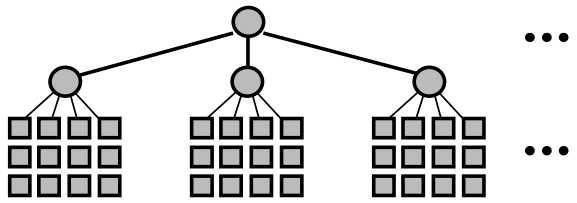


Figure 3
Tree Router Architecture

Each simulator is connected to a router. All communication to and from a simulator goes thru the router. Routers have multiple child clients. All routers, except the single router that is the root of the tree, have one parent router. The root router has no parent. Each simulator has exactly one parent router.

The function of a router is to receive data from clients and parent, and forward (send) the data to any clients or parent that have interest. Implementation requires that simulators and routers communicate interest as well as data. A simulator or router maintains the interest set of its parent router. A router maintains the interest set of all of its clients. When a simulator changes its subscription, it sends a modified interest set to its router. If this modifies the interest set of the router, the router sends the modification to its other clients, and its parent. Interest modifications propagate across the router network until all nodes possess the interest set of clients and parent.

When a simulator publishes data, the associated interest set is intersected with the interest set of its router. If the intersection is not empty, the published data is sent to the router. When a router receives data from a client, the interest set is intersected with the interest set of the router's other clients. For each other client, if the intersection is not empty, the data is sent to the client. The same is performed for the router's parent. Given the connectivity, or topology, of a tree, this set of operations tends to minimize communication while ensuring that all simulators receive all data of interest to them in a timely manner.

Mesh Routers

Next we will describe the design of the new mesh router

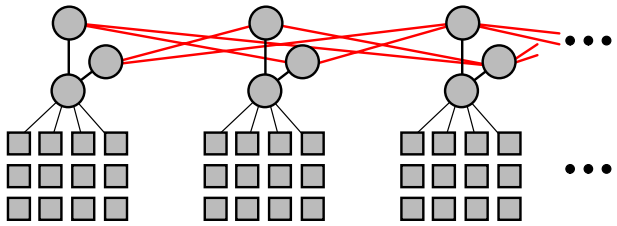


Figure 4
Mesh Router Architecture

The basic communications architecture expands on the original SF Express work, as illustrated in Fig.(2). For purposes of the present discussion, the relevant features of this architecture are as follows:

1. Simulation processors (squares) are grouped with each group associated with a specific router node (“Primary Router”).
2. Message flow from a simulator to its Primary Router had three main components:
   a. Interest Subscriptions: Simulators specify data type of (local) interest.
   b. Data messages up: All messages generated within the Simulator are sent up for possible transmission to other simulators.

c. Data messages down: Messages from elsewhere that match the relevant interest declaration are sent from the router to the simulator.

3. Two additional interconnected layers of router nodes (Pop-Up and Pull-Down) manage all of the interest-screened data communications among the {Primary, Simulators} sets.

4. Strict flow control among the layers prevents communications deadlock, with an additional "token protocol" used to eliminate ineffective data reading attempts.

This architecture was central within the SF Express large-scale simulations, with another class of router-like processors ("Gateways") used to manage interest-screened communications among participating SPPs.

The present effort involves a number of significant extensions from the original SF Express code:

1. Interest enumeration and interest state declarations are now done using tools within RTI-s.
2. Interest declarations are now "two way", involving both interest declarations and publications.
3. Limitations on message size have been eliminated, thus supporting occasional very large "environmental" messages within typical JSAF applications.
4. The entire code base has been reformulated in a rather rigorous object-oriented (C++) form.
5. Communications (along any link in the figure) are now cleanly factored into a number of objects and supported by extensions now incorporated into the RTI-s libraries.
6. The system fully supports mixed communications protocols. Some of the links in the Figure might represent MPI communications while others could be TCP.
7. The Gateway models from SF Express have been reformulated (now essentially clients rather than "special" routers). Taken together with item 6, this greatly facilitates linking of "meta-systems" incorporating LANs and SPP assets.

Performance and resource usage monitoring

Abstraction mechanisms found in many distributed programming systems enhance software reusability and interoperability by hiding the physical location of remote software processes. These abstraction mechanisms, which include HLA's concept of federates (Lightner, 1998) and CORBA's concept of components (Keahey, 1997), greatly reduce the complexity of accessing remote components. But, they come at the cost of reduced visibility, which hinders discovery of faults and impedes understanding of performance characteristics of the distributed system. This section describes a performance and resource usage monitoring tool Monitoring Remote Imaging (MRI) that aids developers in understanding the behavior of HLA simulations by displaying the monitoring data within the context of the execution of the distributed system. Similar specialized tools could easily be envisioned, designed and encoded for other simulations.

MRI displays monitoring data in the context of the federation connection topology. Figure 5 shows the screen dump of a MRI client's resource usage gauges displayed in the context of a three-level tree topology. The large oval pie chart at the top represents the root tree router. The set of rectangles underneath the root tree router represents sub-trees or router subgroups. Each subgroup has a tree router (medium-sized pie chart) connected to a set of federates (smaller pie charts). The first subgroup on the left as only one federate, and the other subgroups have eight federates.

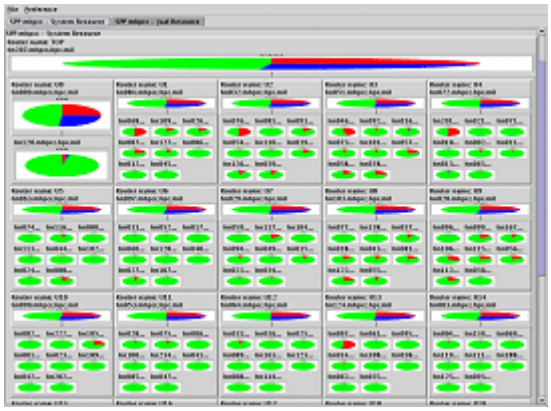


Figure 5
Resource usage data of a JSAF federation displayed within the context of a tree connection topology.

In Figure 5 each CPU pie chart depicts the CPU usage breakdown for one compute node:

- Red for user-level CPU usage
- Blue for system-level CPU usage,
- Green for idle.

Each compute node has two CPUs, but the node is currently only running one process, so typically at most

50% of the CPU is used for non-thread applications like JSAF. At the snapshot when Figure 5 was taken the router nodes within the tree show substantial system-level CPU usage, which indicates the routers are busy accessing kernel-level instructions to send/receive data. The federates in Figure 5 are only lightly loaded. Figure 6 shows alternative XY-plots for displaying time series data. We are currently evaluating the efficacy of the various displays.

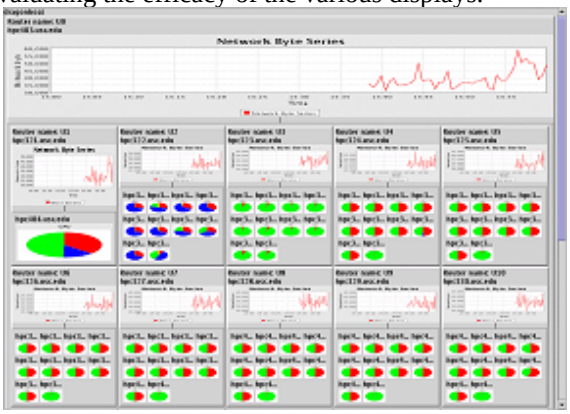


Figure 6
Plotting router network I/O as a function of time

MRI provides a framework for monitoring the performance and resource usage of federations at both at the OS-level and at the application federate level. Performance metric from both levels allows developers to correlate resource usage with JSAF simulation behavior. MRI display clients subscribe to monitoring relay gateways, which periodically push out the monitoring data. This monitoring data is represented in XML for extensibility and flexibility. At the OS-level it monitors the CPU load (user, system, idle), memory usage (user, share, cached, free) and network traffic (packets in/out, bytes in/out). Currently, for such OS-level information MRI uses Ganglia, a cluster monitoring tool from Berkeley's Millennium Cluster Project. At the application level it currently monitors JSAF's internal load, heartbeats, and various types of entity counts (remote, local, ground vehicle). See Figure 7.

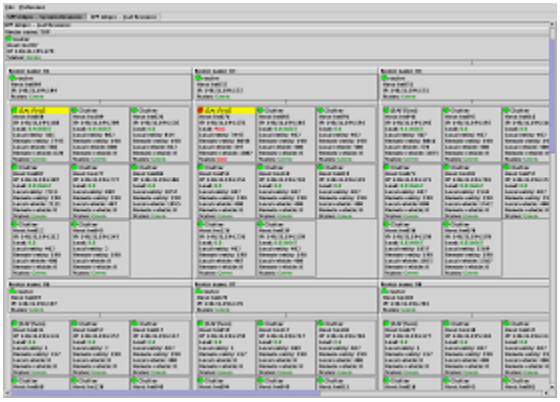


Figure 7.
Custom gauge display for JSAF federates and routers. Green/yellow/red status lights indicate internal JSAF health status. Yellow background

MRI maintains a representation of the federation connection topology in order to generate the gauge displays. This does not violate HLA's information hiding principles of reusability and interoperability, since this topology information is still hidden from JSAF federates, and the federates still have to communicate with each other using HLA RTI's communication infrastructure. The difference is that the connection topology, which is always a vital part of the HLA RTI, is now explicitly represented. Software researchers have argued that explicit representation of software architectures and topologies facilitates better reasoning and understanding [Garlan and Shaw, 1993]. For example, in our case within the context of a topology we can determine the relative importance of node failures/bottlenecks. In Figure 7, node hn068 is highlighted with a yellow background indicating that it failed to emit monitoring data in a timely manner. The failure of node hn068 would bring down the 361 local entities that it is simulating. However, if instead the router node hn084 had failed, then it would have disconnected an entire subtree affecting 6040 entities. If the head router hn207 had failed, then it would result in a forest of disconnect subtrees. Current development is directed at preventing such losses.

Initiation Issues and the "SimPrep Tool"

A major issue when using multiple and geographically distributed SPPs is the effective coordination of intitiation, operation, and termination. There is a large body of research and development literature on various approaches to this issue. (Foster, 1997) While using these existing utilities and tool-kits may perhaps be the smoothest path to an effective implementation, we believe that this is one of the cases where a new tool may be desirable. To illustrate the definition of a need and the implementation of a new tool to serve that need, we will discuss the JESPP "Simulation Preparation" (SimPrep) tool. We do not suggest that it has the broad functionality of a tool-kit like Globus, nor is it suggested that other groups will need or want to develop individual tool-kits in every circumstance.

The preliminary objective of the JESPP exercise is to enable scalable multi-user simulations of synthetic semi-automated battles across multiple SPPs. Accompanying our mission are challenging problems that we must address:

1. Overcoming geographical separation that is inherently problematic in terms of latency, and this experiment is particularly interesting due to the requirements of transporting a large amount of data between the clusters.
2. Accommodating the variation of SPP operational policy, e.g. security policy, software and configuration, and network constraints.
3. Implementing interactive computation in a meta-computing environment. This is a new challenge, and requires a new way of doing business. We need to operate the SPPs in interactive mode, as oppose to their traditional batch-mode model.

1. Trying to juxtapose between ease of use and flexibility. Our GUI application had to be flexible as scripting language scripts. While this challenge is not new to software implementers, they were nonetheless challenges.
2. Having to deal with continuous and large dataset – this along with the need to conduct precise metric. Traditional batch operation on a single or multiple SPPs, while collecting data concurrent to simulations, postpone processing to the post simulation stage.
3. Data collection had to behave as observers and intrude into the collection process, thus be observed.

Solving the challenges above was accomplished against a backdrop of constraints, which included but were not limited to:

The experiment process can be decomposed down to four, disjointed processes; along with accompanying software tools we've developed to facilitate each of the stages:

Stage	Applications
Abstraction stage Designing the network and communication topology, and do simulation preparation.	SimPrep and MARCI collector and MARCI GUI
Implementation stage Deploying our software tools and applications to the SPP compute nodes	MARCI application suite deployed and launched applications
Execution stage Conducting the actual experiment by *game players*	JSAF applications, including tree router, JSAF and ClutterSim.
Analysis stage Studying and analyzing the exercise and performance and effectiveness analysis	MRI and post processing and logger tools

Table 1

During the abstraction stage, we planed and designed the network topology. We were primarily interested in how each of the SPPs would be configured and connected (internally) as well as the network connections (externally) between them. To facilitate this process, which was extremely tedious and error-prone, we developed a software program called *SimPrep* that read in as an extensible configuration (network topology specification) file that utilized PERL programming syntax.

During the implementation stage, we used the MARCI applications to query the clusters for resources. Using the resource information and the configuration file defined (designed) in the abstraction stage, *SimPrep* performed resource allocation and map concrete actual compute nodes to abstract network layout.
There were two output files:

(1) the RID, a flattened connectivity file
(2) a mass launch file.

The RID file was in a LISP dialect and required to be manually stitched into a larger RID file and is understood by the JSAF, clutter, and router applications. The mass launch file was a MARCI specific instruction file on how to launch applications for a specific SPP. Note that the rules for different SPP are specified in the *SimPrep* configuration file.

Once the implementation stage was done, the exercise began. At this point the MARCI application took over. MARCI was responsible for starting and stopping applications – and specifically MARCI along with SimPrep served as the tool with which operators can interface and managed applications on an SPP interactively. This fact contrasted our way of using the SPP with the traditional batch-processing model. The communications between the MARCI GUI and the MARCI collector is a socket-based communication on

top of the SSL Layer and it used public/private key for message encryption.

Our option to use Globus is limited to resource scheduling and resource discovery. We feel that at this stage, as the experiment policy is still be shaped and defined, Globus would be better used when our way of doing business is solidified. We also feel that Globus does not address the conduct of experiment, instead it serves to facilitate the experiment once the rules of engagement have been defined. For future experiments, we feel the Globus will play an important role – especially in the resource scheduling and discovery stage.

Accomplishments and Future Directions

In December of 2002, the JESPP team ran a successful prototype event using a partition of the USC IBM Linux cluster, consisting of some 240 IBM 335 servers, with 2 GHz Xeons, 1 GByte of RAM and both GigE and Myrinet mesh communications. Both the scientists at ISI in California and the operators at JFCOM in Virginia jointly shared control. More than 1,000,000 civilian entities were successfully simulated. They showed appropriate behavior and were stable, even when scanned by the SLAMEM program, emulating two GlobalHawk platforms. To ensure usability and operational validity, about 1,100 warfighting entities were also simulated and controlled in a manner consistent with normal J9 experimentation. Stability and appropriate response to control commands were evident throughout. Several runs were conducted over the course of a week and performance was characterized.

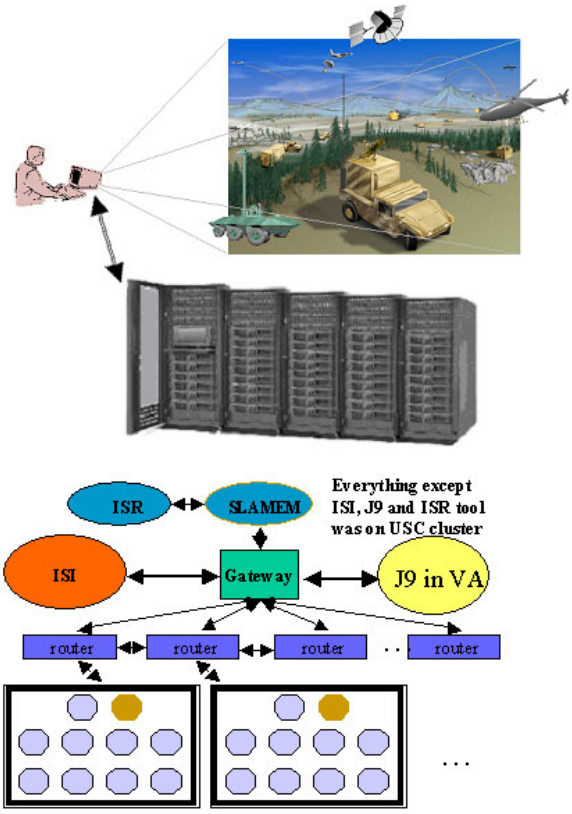


Figure 7
Conceptual diagram of December Prototype Event.

Following the December event, it was decided to show the utility of the DoD's SPP assets by using two Linux clusters, at two High Performance Computing Modernization Program sites. Two centers agreed to support this activity, the Aeronautical Systems Center (ASC) in Ohio and the Maui High Performance Computing Center (MHPCC) in Hawai'i. Maui had the larger resource in this case, a several hundred node IBM Linux cluster with Pentium III processors running at 933 MHz and with 1 GByte RAM per node. ASC's cluster was smaller, but exhibited similar processing parameters. With assistance from the HPCMP PET program, the Defense Research and Engineering Network (DREN) was used to interconnect MHPCC, ASC, and the Joint Forces Command in Virginia. Scalability and stability were recorded. Initiation and system configuration issues were studied and addressed.

The group contributing to the JESPP project has made several noteworthy advances in high performance computing. We note the two-router designs, both of which merit further testing and use. Also, a fresh look at performance monitoring on heterogeneous and geographically dispersed SPPs has yielded a robust

and useful tool that both generates data and presents status information in a visual manner that is useful for both parallel processing experts and simulation professionals. Some unique initiation problems have resulted in a new approach to complex synchronization issues not adequately addressed by either the SAF family software or by more general meta-computing tools.

Open issues for future work:

There is much to be done, of course, in terms of instrumenting and analyzing the existing system, contrasting performance with that from communications options within the current RTI-s baseline. The more interesting studies here will involve comparisons of new qualitative features of the underlying simulations. An example here is the difference between "reduced capability" and "self-aware" clutter (i.e., do clutter objects interact).

Many of the more interesting near-term development paths can be characterized in terms of "special purpose gateways" (now supportable in view of the reformulated Gateway models). Examples include:

- Translation Gateways: Processors to interpret and convert interest declarations among simulations (federates) that do not use a common interest-enumeration protocol.
- Visualization Gateways: Processors (quite possibly multi-processor collections) to request, collect, process and simplify (e.g., iconify) visualization data within very large simulations. (Current model does most of this work within the visualization workstations, giving rise to ample opportunity for death by communications overload.)
- Input Gateways: The "Collect, Preprocess, Summarize" objectives of the Visualization Gateway could be extended to other processes interested in large subsets of the simulation entities. An important example here is SLAMEM.

That is:

> This is not "merely" a translation of existing (i.e., RTI-s) communications procedures. This is the first of a number of steps to qualitatively new capabilities that follow from:
>
> 1. The scalable communications capabilities of the basic architecture.
> 2. The additional capabilities of the "intelligent gateways" supportable within this architecture.

Acknowledgements

This work was directed and funded by the Joint Experimentation Group at the Joint Forces Command and the authors wish to thank Gen. Dubik, Dan Franken, and Anthony Cerri. We especially would like to acknowledge the direction, support and counsel of Rae Dehncke who has been the program manger and guiding light for this project. The authors would like to acknowledge the excellent technical support provided by the members of the Computational Sciences Division of the Information Sciences Institute, Gene Wagenbreth and John Tran, as well as the guidance and assistance from members of the Scalable Systems Division there, the Dr.s Ke-Thia Yao and Robert Neches. Also contributing greatly were the scientists at OTCI, Dr. Tom Gottschalk, Mike Boughton and Marvin Shannon. None of this could be accomplished without the help of the JSAF team Dr. Andy Ceranowicz, Mark Torpey, Bill Hellfinstine and many others. This material is based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

References

Ben-Ari, E. (1998). Mastering Soldiers: Conflict, Emotions and the Enemy in an Israeli Military Unit. New Directions in Anthropology, V. 10. Oxford: Berghahn Books.

Brunett, S., Davis, D., Gottschalk, T., and Messina, P., (1998), Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments, Seventh Heterogeneous Computing Workshop, Orlando, Florida

Cebrowski, A.K., & Garstka, J.J., (1998), Network Centric Warfare: Its Origin and Future, Naval Institute Proceedings, 124/1, 28-35.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J., (2002), Reflections on Building the Joint Experimental Federation, Proceedings of the 2002 I/ITSEC Conference, Orlando, Florida

Foster, I. & Kesselman C., (1997), Globus: A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, 11(2): 115 –128

Garlan, D. and Shaw. M., (1993), An Introduction to Software Architecture: Advances in Software Engineering and Knowledge Engineering, volume I. World Scientific Publishing,.

Hill, R. W., Gratch, J., & Rosenbloom, P.S., (June, 2000). Flexible Group Behavior; Virtual Commanders for Synthetic Battlespaces. Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain.

Keahey, K.& Gannon, D., (1997), and PARDIS: A parallel approach to CORBA, Proceedings. The Sixth IEEE International Symposium on High Performance Distributed Computing, pp: 31-39, Portland, Oregon

Lightner, G.; Zeswitz, S.; & Graffagnini, J., (1998), Practical insights into the process of extending a federation-a review of the High Level Architecture Command and Control Experiment, Proceedings, 2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications, pp: 41-51, Montreal, Quebec

Messina, P. C., Brunett, S., Davis, D. M., Gottschalk, T. D., (1997, April) Distributed Interactive Simulation for Synthetic Forces, In J. Antonio, (Chair), Mapping and Scheduling Systems, International Parallel Processing Symposium, Geneva, Switzerland.

Sanne, J. (1999). Creating Safety in Air Traffic Control. Unpublished doctoral dissertation, Institute of Tema Research, Linköping University, S-581 83 Linköping, Sweden.

van Lent, M. & Laird, K., (1998). Learning by Observation in a Complex Domain. Proceedings of the Knowledge Acquisition Workshop, Banff, Canada.

Appendix E

Supporting Distributed Simulation on Scalable Parallel Processor Systems

Richard Williams
BMH Associates, Inc.
Norfolk, VA
Williams@bmh.com

John J. Tran
Information Sciences Institute, USC
Marina del Rey, CA
jtran@isi.edu

ABSTRACT

The Distributed Continuous Experimentation Environment (DCEE) is a permanent simulation infrastructure being set up by U.S. Joint Forces Command (JFCOM) to support Joint experimentation. DCEE will combine simulations running on both local JFCOM networks and Scalable Parallel Processor (SPP) networks. JFCOM has been working to develop tools to manage a large number of simulation computers with a minimal number of technical support personnel. These tools allow an operator to start and stop various applications, monitor and graph machine resources, generate simulation routing topologies, check network connectivity, and perform these functions in a secure environment.

As DCEE planning continues, the requirement for centralized federation control becomes obvious. The challenge of remotely coordinating the operation of hundreds or possibly thousands of simulation applications looms ever larger. The fact that numerous machines may exist on remote networks further complicates this issue. As an integral element of DCEE, centralized control will need to be expanded to manage and monitor SPP networks along with existing systems.

This paper will address the complex challenges of controlling and monitoring an extensive simulation environment. The paper will introduce the environment, describing the simulations and the SPP. The paper shall also discuss the operational and technical advantages using a centralized set of tools. The paper will not only examine the challenges encountered by attempting to run simulations on SPP networks, but also how these challenges are met.

ABOUT THE AUTHORS

Richard Williams is a Software Engineer with BMH Associates, Inc., supporting Joint Forces Command (JFCOM) in Suffolk, Virginia. He received a B.S. in Computer Science from the University of Central Florida. He has supported federation development for Attack Ops 00 (AO00), United Vision 01 (UV01), and Millennium Challenge 02 (MC02). He is currently supporting work for the Joint National Training Center (JNTC) and Distributed Continuous Experimentation Environment (DCEE).

John J. Tran is a Computational Scientist at the Information Sciences Institute, University of Southern California. He received both his BS and MS Degrees in Computer Science and Engineering from the University of Notre Dame, where he focused on Object-oriented software engineering, large-scale software system design and implementation, and high performance parallel and scientific computing. He has worked at the Stanford Linear Accelerator Center (SLAC), Safetopia, Inc., and Intel Corporation. While at SLAC, he was part of the E&M research team, whose research focuses on solving Maxwell's electromagnetic equations. While at Safetopia, he worked on a high performance encryption file-system, multi-processing communication abstraction layer, and distributed computing and dB interface. At Intel he was responsible for a software tool that performs noise analysis using Intel's circuit design methodology. His current research centers on Linux cluster engineering, effective control of parallel programs, and communications fabrics for large-scale computation.

Supporting Distributed Simulation on Scalable Parallel Processor Systems

Richard Williams
BMH Associates, Inc.
Norfolk, VA
williams@bmh.com

John J. Tran
Information Sciences Institute, USC
Marina del Rey, CA
jtran@isi.edu

INTRODUCTION

The Distributed Continuous Execution Environment (DCEE) is a permanent infrastructure being set up by the Joint Force Command (JFCOM) to support Joint Experimentation. The DCEE will merge distributed simulation systems running at JFCOM with simulations running on Scalable Parallel Processor (SPP) Networks. The reason for merging these systems is to add flexibility in the allocation of resources for use in very large scale simulations.

The government owns a number of SPP systems. These systems represent a very large resource that has yet to be regularly tapped into an interactive computing environment. The SPP systems typically have a number of strengths, such as high speed networks and high quality machines, which make them very desirable for the simulation community. However, they are sufficiently different from the typical simulation environment to present new and difficult challenges.

The Synthetic Forces (SF) Express (Burnett 1997) project recognized that increasing the size and complexity of distributed simulation greatly increases the importance of resource scheduling and allocation. The problem of preparing distributed systems to support simulation is simple to understand and tedious to resolve. Very large simulations can easily become impeded by details. Setup, configuration, installation and execution can easily become difficult tasks if a well designed system is not in place. This paper describes the system we have created to address these issues.

SPP Background and Motivations

The use of the Joint Semi-Automated Forces (JSAF) application on SPPs has its roots in the SF Express project (Burnett, *et al*, 1998) that achieved major milestones in terms of escalating simulation entities counts to an unprecedented level. At the same time SF Express laid down a foundational proof-of-concept that simulation experiments can be conducted over a wide area network (WAN). Operationally, the earlier SF Express experiments spanned across multiple SPPs and were semi-automated; in particular, the resources for each of the experiments were dedicated (reserved) for each scheduled simulation event.

A natural progression that builds on the earlier successes of the SF Express project includes: (1) further increasing the entity counts to match realistic synthetic operational theaters and urban environment, (2) further automate the process and management of simulation events, and (3) further optimize and enhance communication and dataflow by increasing flexibility and network connectivity abstractions.

The modernization of SPP resources, such as the increase in readily available bandwidth between SPP sites (DREN 2003) and the proliferation of freely available OpenSource operating system software (Linux) running on commodity hardware (Intel-based x86 architecture), have helped us increase high performance computing capabilities. The Joint Experimentation Scalable Parallel Processor (JESPP) team set new entity record counts, exceeding the million clutter entity count, with our experiments at the USC cluster in December 2002.

As previously mentioned, the increase in resources (especially those spanning multiple sites) necessitates that the event organizers automate the setup and simulation event as much as possible.

In the earlier implementations of JSAF applications, the communication layer is based on a broadcast model which for inter-SPP communications is not at all possible. Thus, design and implementation network topology abstraction serves as a means to further optimize intra-SPP and inter-SPP communication flow.

Participating Sites and Configuration

The SPP configuration for the Maui High Performance Computing Center (MHPCC) and

Advanced Simulation Center (ASC) are both Linux clusters with dual processor P3's with one to two gigabytes of memory and approximately ten gigabytes of local scratch disk. Both operational sites at the Information Sciences Institute (ISI) and JFCOM run JSAF applications and our controlling/monitoring tools on Linux workstations. The connection between ASC and MHPCC is a high-speed DREN network with sustaining bandwidth of 40 megabits/second (see Figure 1).

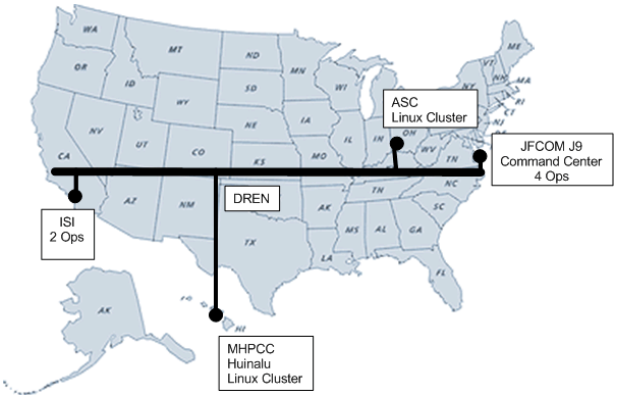


Figure 1 - SPP and operations centers

As noted, the multiple SPPs integrate to form a large simulation system with each SPP being a self-contained cluster of compute nodes. We strive to keep the framework of operation within each individual cluster consistent, despite varying policies amongst clusters.

The SPP Federation

The federation we are trying to support for the SPP testing is comprised of a number of federate applications bound together by a communication protocol called the Runtime Infrastructure (RTI) (Kuhl 1999).

The Federates

Currently there are a small number of applications being tested in the SPP configuration. These include JSAF, Clutter, and Simulation of the Location and Attack of Mobile Enemy Missiles (SLAMEM). These simulations are used to produce entity-level platforms which interact to produce a high quality synthetic environment.

JSAF is a high-fidelity entity level simulation, that can be used in a front-end/back-end configuration where the front-end (or GUI) will be on a local machine, while the back-end will be on the SPP. Entities can be instantiated on the front-end, by either an operator or from a file, but the actual simulation will be on the back-end within the confines of the SPP. This s done to limit the application-to-application communications within the confines of the SPP and to minimize Wide Area Network (WAN) traffic. JSAF also can be used in a "Pocket" configuration where both the instantiation and simulation of entities will be within the confines of a single local machine.

Clutter is a simulation used to add very large amounts of low-fidelity civilian and military traffic to a simulation to confuse sensors and create a more realistic simulation. Clutter will publish a large number of entities but only subscribe to a few interactions.

SLAMEM is used to simulate various operational sensors and provide a proper perception of reality. SLAMEM uses ground truth entities to produce a realistic view of what could be captured by sensors within the simulation. This "view" is fed back into the simulation in the form of target tracks.

The Runtime Infrastructure and Communication

The RTI is the common component which the simulations use to communicate across the network. The RTI uses subscription spaces to divide simulation traffic so that federates only receive objects and interactions in which they are interested. For the DCEE, JFCOM is using RTI-s. By using RTI-s, we have been able to modify the methods of Data Distribution Management (DDM) (Helfinstine 2001) to allow for a much larger number of subscription spaces. For example, in a test completed in March, the city of Los Angeles alone was subdivided into 8,000 subscription spaces and the terrain was divided into over 100,000 spaces. In contrast, the Millennium Challenge 2002 federation was limited to approximately 1000 spaces (Ceranowicz 2002). This change was possible only by changing the communication protocol and using a routing application to perform management controls. All applications in the simulation learn about their routing by reading the Runtime Initialization Data (RID) file and communicating accordingly.

Building Software, Distribution and NFS

To maximize simulation time and minimize time needed for setup and configuration, we have found that having a single source for file distributions and builds is best. Troubleshooting applications started on an SPP is difficult. We do not have the option of redirecting output of possibly thousands of

applications across the WAN. Therefore, we attempted to limit the number of variables as much as possible. We have a single administrator build and package the distribution and then distribute software tunneled through a secure shell (SSH).

Currently, we are configured to use a Network File System (NFS) for all applications and RTI files, but terrain information is stored on each node locally, because the network load of loading binaries is minimal and only occurs at startup. In contrast, the paging in of terrain can create a heavy load during execution.

Application Execution

Executing applications for a distributed simulation in an SPP environment is not simply a matter of submitting a batch job and having everything work. Prior to runtime there must be a process of gathering resources, creating a topology, distributing the Runtime Initialization Data, and preparing command lines. To aid in performing these functions, we developed a set of programs which consists of a daemon to run on each node, a collector daemon to run on the head node of each cluster, and a controlling GUI to run at JFCOM. This system was designed to work in conjunction with a system that was already controlling local machines at JFCOM (See Figure 2).

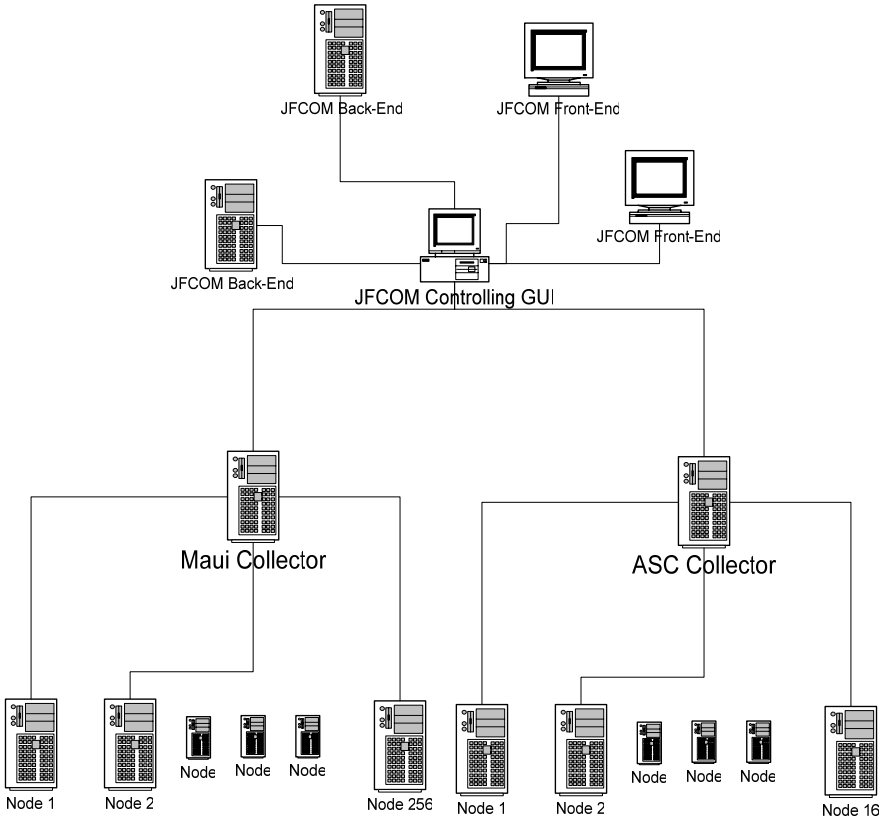


Figure 2 – GUI/Collector/Node Daemon Framework

Starting the Daemons

Generally speaking, each cluster has a job queue that manages resources and schedules job requests [and for the most part, SPP policy disallows interactive login shells.] Our approach is to start the collector on the head node and then submit the node daemons to the job queue (see Figure 3).

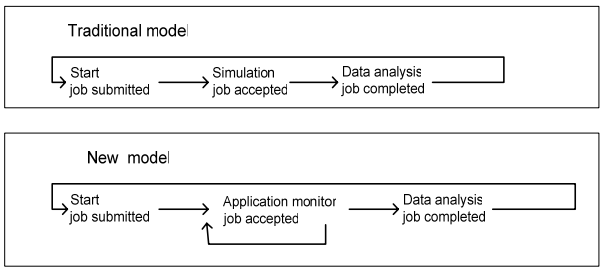


Figure 3 - Comparison of work flow between the traditional batch model and modified batch model.

Once the job is running, using the front-end, we can interactively start and stop applications. If a topology or configuration requires modification, the front-end can automate the workflow so the entire group of cluster applications start and stop with a single click.

New Way of Doing Business

Our methodology adheres to the cluster-job submission policy and provides a flexible interactive (and automated) interaction between the operator and the SPPs. This new approach warrants a brief discussion on this new way of doing business. Figure 3 compares the traditional method of interacting with clusters to the new approach. This approach provides real-time control interface that can often optimize resource utilization and provides (retains) a semi-automated model for rapid start/stop of applications.

Furthermore, this model is relatively fault tolerant. Despite previous efforts to make SPPs more fault tolerant (Squyres, 2000), the traditional model has some drawbacks. For example, if a node or an application dies, that node is lost. To restart applications on that node, users must submit new job submission requests. Using the interactive SPP model, operators can restart dead applications in real time, or even reconfigure the cluster topology, without having to start over from scratch.

Gathering Resources

Discovering resources on multiple SPP systems presented some problems. The first problem was trying to interact with machines on a disparate network. We opted to run a collector daemon on the head node that could be contacted by both applications on external networks and daemons that run on the internal nodes. All commands to nodes on a cluster would then be tunneled through the collector. This collector maintains a list of all nodes, and when requested by an approved external application, it provides a status of the nodes.

A second challenge that the SPP environment presented was the absence of a broadcast/multicast capability. This meant that to recognize existing machines on the cluster, the nodes would have to be configured with the IP address of the collector prior to execution. When the node daemons start up they attempt to connect to the collector to pass it their address information. Thereafter, the daemons send a heartbeat signal once per minute, while the collector gathers information on the node resources and waits for tasking from the external controlling application.

Within the heartbeat is information regarding free memory, CPU type, and load. This information is used to give the resources a rating that represents the expected simulation capabilities of the given node.

Abstracting Network Connectivity with Topology

Because broadcast messaging does not scale well as resources increase, RTI communication with the SPP is restricted to point-to-point (or a subset of broadcast, instead of global broadcast). Formal organization of the communication topology maps applications to resources. Two connectivity maps are currently being implemented and studied: (1) tree-based topology (Hellfinstine, 2001), and (2) mesh-based topology (Brunett, 1998). We have successfully tested the tree-based topology in current and past experiments, and we intend to pursue in a near future the implementation of the mesh-based topology.

Based on network load observation and metrics collected during test trials, we designed each topology instance. A topology is always conceived before simulation and is represented by a suite of perl scripts that generate RID files containing the connectivity maps.

Creating a Mass Launch File

Following the topology generation process, we know which application will be running on a given resource. The next step is to create a file to store the parameters specific to each application instance. This file generation is an automated process which produces a text output that can be read by the cluster controlling application.

Operators can then cut, copy, paste, and modify entries in the mass launch files as necessary. Multiple files can be generated to support multiple launch configurations.

JSAF back-ends require a special procedure. Each JSAF back-end is assigned a Persistent Object (PO) database number, which enables control from a front-end with the same database number. This number is usually shared between multiple back-ends to allow the entity load to be distributed among multiple machines. Grouping PO databases with topology groupings lessens traffic across the routing applications and keeps objects and interactions to a local set of machines.

Executing a Mass Launch

Once the Mass Launch File is completed, the system should be ready for launch. The operator of the controlling application then selects the tasks to be performed, enters the exercise name, selects the proper terrain and hits the launch button. The collection of tasks is then sent to the corresponding collector, which in turn delegates to the proper nodes.

To handle the large number of tasks potentially given to the collector, a task queue was created for prioritization, future scheduling of tasks, and for the limiting of simultaneous connection threads to ten to prevent overwhelming the system. During a test in March, we were able to launch 240 applications on as many nodes within 60 seconds.

After receiving the application execution task from the collector, the node starts up the application and sends out a heartbeat. The collector receives the heartbeat which should indicate the application has started. At this point, the controlling GUI's operator can update the status to ensure all applications started up properly. As the applications start, they can be observed joining the federation by executing RTI print commands within the parser of any local federates.

Problems with Running on a Cluster

Typically when running simulations in a Local Area Network (LAN) environment, we run an application in the GNU Project Debugger (GDB) and direct the output to either a local monitor or a central monitoring station. This allows us to obtain stack traces and interact with any applications as necessary. Because SPP machines have no displays and redirecting a display across the WAN is not an option we are unable to easily monitor output. This adds difficulty to troubleshooting and difficulty in resolving configuration issues. We do log output to a file, but this is much less useful than an interactive debugging session.

Killing Apps / Restarting the Simulation

Once set up, the federate and routing applications can be started and stopped as necessary to support the needs of the federation. Stopping the simulations is accomplished in a similar manner to launching the simulations. The operator selects the tasks that were launched and simply presses a button to bring down the selected applications. As in the launch, tasking from the collector to the nodes is controlled to prevent overloading the systems. Operators can also select individual nodes to restart if desired. This option might be necessary if a specific application or group of applications become unresponsive.

SPP Resource Monitoring

The monitoring of SPP resources was recognized early on to be a necessity in any simulation support system. Specifically, we wanted to graph memory usage, load average, and network statistics on individual nodes. We needed to monitor these parameters in real time to analyze how changing parameters, code, topologies, and the many other variables associated with the simulation were affecting the individual machines.

We also wanted to ensure that any monitoring design did not create an unnecessary load. We decided to put monitoring capabilities into the same node daemon, collector, and application GUI framework that had been designed for the application execution system. This allowed the operator to query the resources for information by going through the collector

Analyzing Memory/CPU Utilization

We decided to embed the load average and memory usage information in the heartbeat from the node daemon to the collector. For these parameters, the once a minute sampling rate seemed sufficient and gave a good snapshot of the status of the nodes. We found that graphing the information in relation to time provided the most useful information by providing insight into trends.

We also wanted to ensure that any additional factors we wanted to graph or analyze could be added without much work. Efforts were made to generalize

the graphing and data gathering capabilities so that additions could be made easily.

Analyzing Network Statistics

Network information was not embedded in the heartbeat because the once-a-minute reporting did not always suit our needs. Quite often, a five-second polling interval could catch spiking data that a sixty-second interval would not.

To accomplish this, we had the application query the collector with a list of nodes to be analyzed. The collector then queried the node daemons for network information which could then be used to create or update an existing graph. Using this method the operator can select any update rate he or she desires. Since this process generates much heavier load than obtaining memory or load information, we would typically limit the nodes we would graph to a small set.

Expected Problems and Bottlenecks

Each simulation we support has unique limitations. In benchmark testing, we observed that a JSAF using RTIs consumed about 20 KB of memory per entity subscribed or published. This meant that a JSAF application on a machine with 1 GB of RAM could be aware of about 40,000 entities before memory swapping occurred. This number is not definitive due to a variety of other factors, including terrain paged in and other applications loaded.

Clutter only subscribes to some of the interactions and does not subscribe to remote objects at all. This means that memory becomes less of an issue and that the CPU becomes the primary limiting factor on number of entities the application can simulate. Figures 5 and 6 show CPU and memory information graphed from a machine on a mega-sim cluster node running a clutter application.

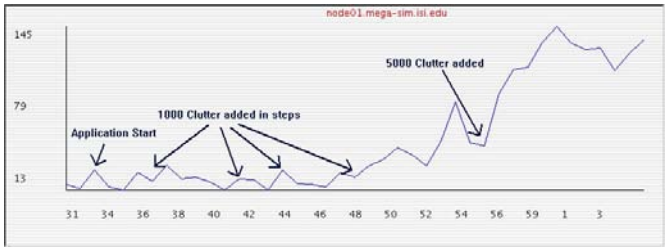


Figure 5 - Clutter CPU Load (Y axis) over time (X axis)

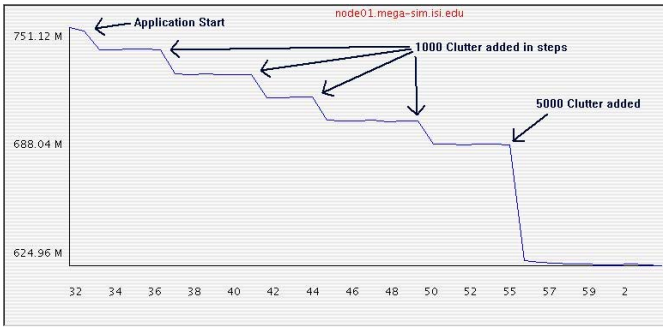


Figure 6 - Clutter Free Memory (Y axis) over time (X axis)

Routing applications on the head nodes of the clusters were the first location where network bottlenecks were expected to appear. When an application, or group of applications, over-subscribes to data that is then requested to go across the WAN, available network bandwidth can easily be exceeded. Figure 7 shows the network statistics on the head router on mega-sim. For the test, a JSAF at a remote site subscribed and unsubscribed to clutter on a cluster node. Graphs such as this can identify problems in DDM, machines, and networks.

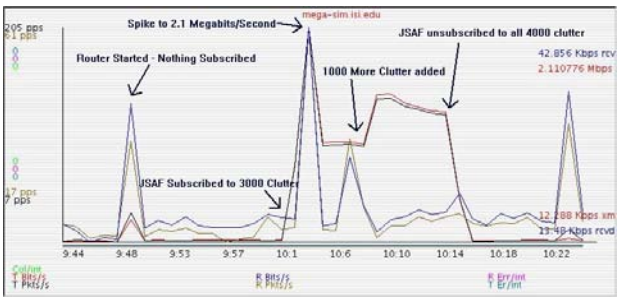


Figure 7 - Head router network statistics (Y axis) over time (X axis)

Although subscription spaces may be divided into very small areas, individual entities or groups of entities can still have subscription regions that overlap a large number of spaces, creating a large amount of traffic. For example, during a test in May 2003, we simulated approximately 100,000 clutter vehicles in the Los Angeles area on the Maui SPP. We then flew an F-16 (simulated at JFCOM) into the area. The F-16 had a subscription radius of approximately 30 nautical miles and subscribed to a large portion of the 8000 spaces in the area. This quickly overwhelmed both the local JSAF and pushed the network limits as well. While this type of effect was expected in this test, it demonstrated that a simulation operator can easily perform a seemingly insignificant action which will have catastrophic consequences on the federation. By adding

monitoring capabilities to the routers we will better be able to catch these issues before they become a problem and help developers create better mechanisms for interest management.

Running a simulation with a very large number of entities and using dynamic subscription regions increases the risks. The potential to "blow out" any element of the federation exists and requires the support team to recognize and try to predict potential problems. Monitoring must inform the development team how changes in software, topologies, or network infrastructure, all machines throughout the simulation. Through diligent monitoring, methods to ensure the simulation delivers the most information possible while operating reliably can be discovered.

Security

To ensure that only individuals with proper authority are able to execute applications on the cluster from remote sites, we have examined two possible solutions.

SSL

Currently, all commands to the collector from the GUI application are executed through a Remote Method Invocation (RMI) connection via a Secure Socket Layer (SSL) (JSSE 2003). The collector is configured so that only clients with a private key (also known as the keystore) and the proper passphrase can connect. The keystore and passphrase are verified by a public key or truststore kept on each of the collectors. The use of SSL serves two functions: data encryption and authentication. SSL uses public key cryptography to provide authentication, while secret key cryptography and digital signatures ensure privacy and data integrity. The private/public key pair is generated prior to file distribution. The public key is then placed in the distribution, allowing the collector to authenticate connecting clients. The distribution of the private key must be controlled and only placed on machines which are designated to control the cluster.

Kerberos

The High Performance Computing (HPC) officer requires that SPP users have a Kerberos (Tung 1999) access card and that applications running on their clusters adhere to their security policy. We are currently studying options to ensure the experiments follow the strict HPC security guidelines, including:

1. Use of Kerberos tunnels between SPPs and operational centers which would authenticate and encrypt all communications.

2. Re-design the communication layer with the explicit use of the Kerberos library for point-to-point communications.

Future efforts include experiments to analyze the performance impact and validate both approaches.

Conclusion

We have attempted to design a system that gives a single administrator the capability to set up, execute, and monitor multiple simulation applications, on possibly thousands of machines, on disparate networks. We have also stressed simplicity and minimized the number of steps required to run the simulation. We believe that we have succeeded to a great degree.

The application execution system alone should save countless hours by simplifying a process that had been extremely obtrusive. By getting away from the traditional SPP batch model we have added the ability to bring applications up and down at will. Further, the monitoring capabilities we have added to the system should allow us to predict and recognize problems that may degrade the simulation. All the tools we have created are not simply nice to have, they are a necessity if we wish to run interactive simulations in an SPP environment.

Difficulties will continue to arise when dealing with systems that we do not own. These difficulties that must be addressed include: adhering to other organization's security rules, resource scheduling restrictions, and system management procedures. In addition, future simulation efforts, though dealing with systems that are complex in nature, should be simple as possible to use.

ACKNOWLEDGEMENTS

The authors would like to thank Andy Ceranowicz, Bill Helfinstine, Steve Bixler, Rae Dehncke, Jason Boyer, Phillip Amburn, Ken Hornstein, Nicholas Pellegrini, Gene Wagenbreth, Ke-Thia Yao, Dan M. Davis, Robert F. Lucas, George Thompson, and all the various site support personnel for their help in various aspects of this project. This material is based in part on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is

authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Burnett, S., Davis, D., Gottschalk, T., Messina, P., Kesselman, C., (1997), Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments. Retrieved April 15, 2003 from ftp://ftp.globus.org/pub/globus/papers/sf-express.pdf

Burnett, S. and Gottschalk, T. (1998). "A Large-Scale metacomputing frame for the ModSAF real-time simulation." *Parallel Computing*. Elsevier, Amsterdam.

Ceranowicz, A., Torpey, M., Helfinstine, B., Evans, J., Hines, J., (2002), Reflections on Building the Joint Experimental Federation.

DREN webpage. http://www.hpcmo.hpc.mil/Htdocs/DREN/dren-def.html

Helfinstine, B., Wilbert, D., Torpey, M., Civinskas, W, (2001), Experiences with Data Distribution Management in Large-Scale Federations, Simulation Interoperability Workshop, 01F-SIW-032, Sept 2001.

JSSE Reference Guide http://java.sun.com/j2se/1.4.1/docs/guide/security/jsse/JSSERefGuide.html

Kuhl, F., Weatherly, R., and Dahmann, J., (1999), Creating Computer Simulation Systems: an Introduction to the High Level Architecture. Prentice Hall.

Squyres, J.M., Barrett, B., Lumsdaine, A. The System Services Interface (SSI) to LAM/MPI. Technical Report TR575, Indiana University, Computer Science Department.

Tung, Brian (1999). Kerberos: A Network Authentication System. Addison Wesley.

Appendix F

21st Century Simulation: Exploiting High Performance Computing and Data Analysis

Dan M. Davis
Information Sciences Institute, USC
Marina del Rey, California
ddavis@isi.edu

Garth D. Baer
Oracle Corporation
Culver City, California
garth.baer@oracle.com

Thomas D. Gottschalk
California Institute of Technology
Pasadena, California
tdg@cacr.caltech.edu

ABSTRACT

This paper identifies, defines, and analyzes the limitations imposed on Modeling and Simulation by outmoded paradigms in computer utilization and data analysis. The authors then discuss two emerging capabilities to overcome these limitations: High Performance Parallel Computing and Advanced Data Analysis. First, parallel computing, in supercomputers and Linux clusters, has proven effective by providing users an advantage in computing power. This has been characterized as a ten-year lead over the use of single-processor computers. Second, advanced data analysis techniques are both necessitated and enabled by this leap in computing power. JFCOM's JESPP project is one of the few simulation initiatives to effectively embrace these concepts. The challenges facing the defense analyst today have grown to include the need to consider operations among non-combatant populations, to focus on impacts to civilian infrastructure, to differentiate combatants from non-combatants, and to understand non-linear, asymmetric warfare. These requirements stretch both current computational techniques and data analysis methodologies. In this paper, documented examples and potential solutions will be advanced. The authors discuss the paths to successful implementation based on their experience. Reviewed technologies include parallel computing, cluster computing, grid computing, data logging, OpsResearch, database advances, data mining, evolutionary computing, genetic algorithms, and Monte Carlo sensitivity analyses. The modeling and simulation community has significant potential to provide more opportunities for training and analysis. Simulations must include increasingly sophisticated environments, better emulations of foes, and more realistic civilian populations. Overcoming the implementation challenges will produce dramatically better insights, for trainees and analysts. High Performance Parallel Computing and Advanced Data Analysis promise increased understanding of future vulnerabilities to help avoid unneeded mission failures and unacceptable personnel losses. The authors set forth road maps for rapid prototyping and adoption of advanced capabilities. They discuss the beneficial impact of embracing these technologies, as well as risk mitigation required to ensure success.

ABOUT THE AUTHORS

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute, University of Southern California, and has been active for more than a decade in large-scale distributed simulations for the DoD. While he was the Assistant Director, Center for Advanced Computing Research, California Institute of Technology, he managed Synthetic Forces Express. He has also served as a Director at the Maui High Performance Computing Center. He served in the USMC on active duty and is a Commander, Cryptologic Specialty, U.S.N.R.-Ret. He has been the Chairman of the Coalition of Academic Supercomputing Centers and received a B.A. and a J.D., University of Colorado, Boulder.

Garth D. Baer is a technical analyst who is studying the impact of policy on technology as well as the changes technology makes on policy formulation and implementation. A Principle Support Engineer at Oracle Software Corporation, he develops new web-based applications and troubleshoots production database issues. Earlier, he was a Mission Control Engineer for Milstar Communications Satellites at Lockheed-Martin. He recently was invited to join a multi-university group developing a vision statement for DoD policy on M&S. He received Bachelors in Physics, Univ. of Colorado, Boulder and a Masters in Technology Management, Colorado Technical Univ.

Thomas D. Gottschalk is a Lecturer in Physics at the California Institute of Technology and a Member of the Professional Staff, Center for Advanced Computing Research there at Caltech. For the last decade, much of his research has been on the use of parallel computers to simulate various physical phenomena. His instructional duties include his upper division course on Statistics for Caltech Physics Graduate students. He received a B.S. in Physics from Michigan State University and a Ph.D. in Theoretical Physics from the University of Wisconsin.

BACKGROUND AND INTRODUCTION

This paper will discuss the need to augment simulations, offer enhancements, and show how these enhancements can be implemented, using the authors' JFCOM/Urban Resolve experience as examples.

The Need to Improve Simulations

For thousands of years, leaders have used various representational methods to prepare for the defense of their societies. These have ranged from the venerable game of chess to complex electronic emulations of combat. When threatened, there is an understandable pressure to use what has proven reliable in the past and there is a countervailing desire to make use of effective new techniques. Two of the promising technologies available to defense leaders today are high performance parallel computing and advanced data analysis.

Contemporary analysts are faced with increasing pressure to provide more opportunities to both analyze the present dangers and train for the future operations. The vacant battlefield of yesterday is being replaced by the crowded urban warfare environment of today, populated with non-combatants for whom there is an increased sense of responsibility. Weapons of increased destructive power and refined targeting capabilities make it both possible and necessary to honor this sensitivity. Planners and trainers must have access to simulations of unfettered scale that are built on increasingly sophisticated environments, with better emulations of foes and more realistic civilian populations. The coordination and synergy of these simulation and analytical capabilities are necessary to deliver insights for the analyst and trainee.

There are well-recognized limitations that restrict the full exploitation of what the DoD calls Forces Modeling and Simulation (FMS), (HPCMP, 2004). This paper focuses on two:

- The inherent constraints of current computer-use paradigms
- The restrictions found in traditional techniques of data analysis

In order to meet the two-fold test of reliability and efficacy, new capabilities designed to overcome these limitations must provide sufficient improved utility to warrant the risk and effort expended in adopting them.

The nature of the adoption process is critical. The correct approach will lead to early productivity, low risk and continued utility. A well-thought-out plan, following the proven paths of analogous analytical disciplines, will reduce cost and accelerate benefits. Disciplines of interest include academic research fields investigating physical and biological phenomena. After several decades of using high performance parallel computing and advanced data analysis techniques in these areas, the pitfalls to avoid and the productive paths to follow have been clearly established.

Limitations Imposed on Modeling and Simulation by Current Computing Paradigms

The FMS community has become accustomed to waiting for the additional power represented by the doubling of circuit devices on a computer chip every 18 months. Being able to move from the floor of the gym at the Naval War College (see Figure 1*)* onto the vastly larger canvas of a digital computer terrain database was a momentous leap.

Figure 1. *1930's - U.S. Naval War College personnel conducting simulated campaigns on a gym-sized floor.*

The more distant horizons, such as global-scale, high-resolution terrain environments, seem out of reach. The FMS community has an opportunity to overcome this unnecessarily limited vision.

Terrain databases are now available in multiple resolutions and for nearly every area of the globe. Using workstation and PC technology hosted on LAN configurations, truly incredible advances have been made in our ability to provide a realistic and geographically appropriate environment for conducting large operations (Ceranowicz, 2002). Even these capabilities, however, are often limited in two important dimensions: resolution and total area. As the areas of interest broaden for both the analysts and policy makers, the need to have access to representations of any terrain becomes more imperative.

An example of an important feature of current computer practice to which the community has become accustomed is the constraint imposed by the limits of individual processing speed. The desire to represent sophisticated behaviors requires ever-increasing processor power, and this is magnified by the desire to run multiple instances of non-deterministic simulations to evaluate the range of outcomes (Horne, 1999). The need to represent tens of thousands of entities that are "aware" of each other also impacts performance. In one class of this "awareness," entities are within a range where they can see each other. A much more extreme case is now of concern: the high altitude intelligence platform with sensors that can "see" virtually every entity in an entire theater of war. Current programming, as exemplified in the SemiAutomated Forces (SAF) programs, handles this location and awareness issue by running an inter-visibility calculation every few milliseconds. Obviously, with a huge number of entities, this represents a huge compute burden. Current practice shows this type of situation can be simulated on a typical single processor of present-day (2004) capacities at only a few hundred vehicles. In Millennium Challenge 02, a network of similar PCs on a LAN, experience seems to indicate that the total vehicle count is limited to a few tens of thousands - not enough for military vehicles (Ceranowicz, Dehncke &Cerri, 2002).

Moreover, modern battlefields are rarely located on remote plains, and the battles in urban areas are not fought with the destructive abandon of World War II, as in Stalingrad. Instead, the modern analyst is looking for ways to achieve national goals while operating in populated urban areas with no loss of non-combatant life, minimal destruction of civil infrastructure, and reduced losses to friendly forces. For that reason, the simulations-enabled analyst is faced with the challenge of trying to understand how modern intelligence platforms can view a city full of vehicles and other entities. Clearly, something on the order of a million civilian entities approaches realism; a few thousand does not.

Limitations Imposed by Traditional Data Analysis

Similar constraints are observed when using only the traditional methods of data analysis. Historically, the validation of the insights gained from simulation are not infrequently lost by virtue of the imposition of accepted views.

Analytical approaches have not changed much over the intervening decades. With all of our increased sophistication in electronically produced simulations, one very common method of strategic deliberation remains the observation, logging and analysis of simulation outcomes by subject matter experts (SMEs). The authors maintain that adopting and implementing analytical techniques used in the behavioral sciences and operations research should enable these experts to be even more valuable.

In trying to understand the output of simulations similar to Project Albert, one is faced with a virtual flood of information (Brandstein, 1998). This flood presents problems in collection, collation, and consideration. Many programs are driven by the application of a series of pre-established probability tables for many of their activities, *e.g.* accuracy of fire, damage occasioned by weapons strike, mechanical failures. Against these tables, a random number is applied and the resultant action is implemented. This results in a non-deterministic simulation. Analysis can be much enhanced, if the simulation is run multiple times, with the resultant outcomes appropriately analyzed.

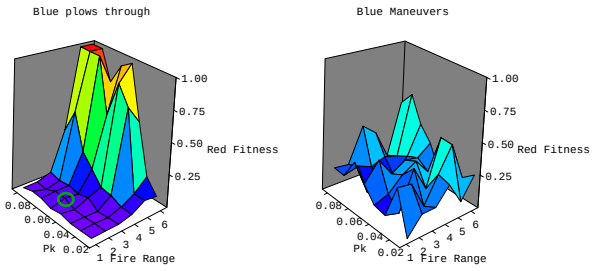


Figure 2 . *Three-dimensional representations of the effects (fitness) of two parameters (fire range and possibility of kills) with maneuver* (Brandstein, 1998).

As the analysts consider the outcomes, they are faced with assessing the impact of several parameters

simultaneously. These multi-dimensional solution spaces become difficult to visualize, as *n* exceeds 3. (see Figure 2) Tabular representations of "killer-victim scoreboards" are not uncommonly seen as insufficient to produce all of the insights that are necessary.

In addition to the data that is collected from running analytical simulations, there is also a huge amount of data that is or could be collected from simulations run for training purposes. This issue of the appropriate and improved use of the data collected for purposes other than its analytical content is treated in more detail later.

Future Needs of Analysts

It could be argued from these few examples that the analysts of today are faced with problems for which current technology implementations are not adequate. While analysts have done heroic duty in developing work-arounds for these limitations, the degree to which they are missing important insights remains unquantified, and disturbing. By analogy, the physical sciences found themselves in a very similar position early in the days of simulations on computers and the subsequent analysis of physical phenomena. Their experience suggests that the current practice of reducing resolution and geographical scope of FMS scenarios is leading in the wrong direction. It may be robbing the analysts of insights that could be extracted from more sophisticated and detailed models running on larger terrain databases.

The FMS community is also increasingly experiencing pressure to simulate some of the more complex forms of human behavior. One area of great interest in military science is the actual role of the individual soldier and commander (Ben-Ari, 1998). More computing power will be required to be able to deliver analytical and training platforms that can model emotion in a useful way (Garlan, 1993).

While FMS analysts now have access to much more computing power than they did a few decades ago, there is evidence that significant additional capacity could be implemented, with a high degree of confidence in its expanded utility, reliability, and stability. It is neither necessary nor desirable to move into the future using untested technology nor is it wise to duplicate already available, useful programs. Others have broken the ground that the FMS community can now till.

HIGH PERFORMANCE COMPUTING

Originally, computer scientists considered that the only way to speed up the process was to accelerate the CPU. At a certain point, however, it seemed obvious that the technology would not be able to keep improving processors to do calculations faster by increasing clock speed. They would similarly not be able to continue to make each clock cycle more effective by adding functions to the processor (Moore, 1965).

That led to theorizing about parallel computation and harnessing more than one computer to work on the same task. A generalized theory of parallel processing effectiveness was advanced by Amdahl, in which he carefully described the speed-up that one would expect (Amdahl, 1967). Starting with work at Caltech on the Intel Delta, with 512 64-bit processors, increasing numbers of simulations have been effectively parallelized for the big machines, with high-speed inter-node communications fabrics. As this size was orders of magnitude greater than the early limits theorized, this class was often referred to a Massive Parallel Processors (MPPs).

Table 1. *World's fastest supercomputers.*

Rank	Site Country / Year	Computer / Processors Manufacturer	R_{max} R_{peak}
1	**Earth Simulator** Japan/2002	**Earth-Simulator** / 5120 NEC	35860 40960
2	**Los Alamos National Lab** U.S./2002	**ASCI Q - AlphaServer, 1.25 GHz** / 8192 HP	13880 20480
3	**Virginia Tech** U.S./2003	**1100 Dual 2.0 GHz G5 / Infiniband / GigE**/ 2200 Self-made	10280 17600
4	**NCSA** U.S./2003	**P4 Xeon 3.06 GHz, Myrinet** / 2500 Dell	9819 15300
5	**Pacific NW National Lab** U.S./2003	**Integrity Itanium2 1.5 GHz, Quadrics** / 1936 HP	8633 11616
6	**Los Alamos National Lab** U.S./2003	**Opteron 2 GHz, Myrinet** / 2816 Linux Networx	8051 11264
7	**Lr. Livermore National Lab** U.S./2002	**MCR Linux Cluster Xeon 2.4GHz, Quadrics** / 2304 Linux Networx/Quadrics	7634 11060
8	**Lr. Livermore National Lab** U.S./2000	**ASCI White, SP Power3 375 MHz** / 8192 IBM	7304 12288
9	**NERSC/LBNL** U.S./2002	**SP Power3 375 MHz 16 way**/ 6656 IBM	7304 9984
10	**Lr. Livermore National Lab** U.S./2003	**xSeries Cluster, Xeon 2.4GHz, Quadrics**/ 1920 IBM/Quadrics	6586 9216

The Top 500 Supercomputers list presents rankings in order of performance using LINPAC, a common benchmark. The top ten of the list for 2003 is reproduced above in Table 1. The number of

processors follows the name (van der Steen, 2003). Note that the least amongst these has 1,920 processors and that the biggest, but not most powerful, has 8,192. This list covers only the supercomputers that are engaged in work that can be publicly acknowledged.

Linux Clusters: The Beowulf Concept

This last cost-based definition brings us to the next important concept: commodity clusters, or Beowulfs. Dr. Thomas Sterling propounded and popularized large-scale parallel computing through the use of cheap commodity components: CPUs, power supplies, RAM, internode communications, operating systems and software (Sterling, 1999). By taking best advantage of the cost benefits of mass production, he collected and organized mass numbers of commodity processors. These are usually Intel architecture PCs, with communications between them using gangs of low-cost Ethernet switches (or the more expensive but more powerful cluster communications switches).

Figure 3 *The IBM Linux cluster at the Maui High Performance Computing Center.*

Beowulfs typically use the largely free operating systems like Linux and the GNU series of compilers (see Figure 3). For internode communications programming, there are a number of languages following the Message Passing Interface (MPI) standard. These may be obtained without paying the expensive license fees that are typical with some of the proprietary supercomputers.

The Beowulf technology is not the most effective one for computing that requires exceptionally high-speed serial computation, exceptional floating-point power, or exceptionally low latencies for their internode communications. Clusters are fortunately very useful for most programs. Unlike the Cray series that were very high-speed vector machines and required liquid cooling, the Beowulf series are now universally air cooled, requiring only sufficient machine room cooling to remove the heat from the amassed processors. The avoidance of the efficient, but expensive, CPU/liquid-coolant interface is an incredible cost savings. A typical price for a significantly sized cluster is on the order of a few thousand dollars per node.

Grid Computing

If processors can be amalgamated to produce more power locally, then there could be even more power made available if remote computers could be similarly connected to provide additional processors. The previously mentioned concept of scalability clearly comes into play and one new concept must be considered. Most of the clusters and supercomputers discussed so far have been homogeneous, *i.e.* all of the processors are the same. If grid computing entails using clusters and processors from different sites, then the likelihood of homogeneity falls rapidly. Fortunately, Beowulfs have been remarkably tolerant of heterogeneity and data will be later adduced to show the capabilities of grids made up of the Beowulf Linux clusters and the proprietary supercomputers.

Grid computing usually conveys the concept of using a Wide Area Network (WAN), frequently the Internet itself, to connect remotely located SPPs, both supercomputers and Linux clusters. The landmark work on this innovation was done by Ian Foster and Carl Kesselman (Foster, 1997). In order for all of these diverse and dispersed assets to be useful, there must be methods of coordinating, initiating, and controlling them. The tool developed by Foster and Kesselman is called Globus and is generally recognized as a very effective way to approach this type of distributed high performance computing.

Another, more localized version of this concept, is that of using all of the idle PCs on an organization's LAN. This involves running processes on the various PCs making those processors available to the central user when they are not in use by the PCs "owner." When the owner interfaces with his computer in any way, it immediately suspends the remote process and redelivers control to the owner. One popular program providing this service is Condor (Litzkow, 1998.) This technique is a natural choice for a type of computing that does not need cycles on demand. One nationally distributed use of this concept is doing signals processing as part of the search for extraterrestrial intelligence.

Parallel Data Handling

The two foci of this paper are parallel computing and advanced data handling techniques in FMS. While much of the issue of data handling is appropriate for the section on data analysis, some portion of it is more closely tied to parallel processing. Parallel distributed processing both enhances and encumbers data collection, storage and retrieval. One of the major daily uses of high performance computing is the rapid processing of huge masses of transactional data by retailers and financial institutions, an indication of its value in this arena.

Like parallel processing, there is an extensive experience base in parallel data handling. At the Information Sciences Institute, a distributed data system has been developed to support the SAF simulations. Client applications communicate using RTI routines and data that is identified is stored on local disks at each node. A central aggregator acts to query the tasks, when desirable, and collects all information at the end of the simulation. The data content is then analyzed and archived. As it is new technology, the techniques for maximizing the utility of the parallel capabilities are not universally practiced, but the expertise is easily accessed.

Figure 4. *Tertiary storage tape silos at USC.*

ADVANCED DATA ANALYSIS

As simulations have moved from the gym floor to the computer, a similar change has taken place in the means of assessing the results of the exercises. When the SAFs were first used to train tank crews, the most important factor was face validity. As long as the tanker trainee perceived the representations as realistic, the simulation was considered to be a success. Now that the community has moved from a few vehicles to more than one million vehicles, the need for a more elaborate approach has become clear. Policymakers and leaders of the simulation community now seek new ways to exploit the data being collected. (Dubik, 2003).

Additionally, this country no longer has the luxury enjoyed in past wars of taking months to mobilize technology for defense efforts, and learning from early combat experience to hone later tactics. Today's battlefield is much more technologically loaded, complex and fast-paced (Cebrowski, 2000). It follows that there is a need for more complex, faithful and illuminating simulations of future battlespaces. The insights needed must be more timely and of greater specificity, in order to defend against new foes who are less identifiable, less predictable and more capable of attacking asymmetrically.

One of the first issues of concern is defining just what the simulation community and government leaders should and can extract from the simulations. Rather than considering this issue *de novo*, much can be learned from the Operations Research approach (Kleijnen, 2001). Many of their techniques have already been implemented on SPPs and their rigorous analysis of critical parameters is very useful.

Advances in Database Technology

As data sets have grown exponentially larger and more complex, so also has the technology grown to query that data and return useful and timely result sets. While the expenditures of the DoD are not insignificant in this field, much of the productive innovation is being delivered out of the commercial database market and much of the intellectual leadership resides on the campuses of the U.S.'s top research universities. Search engines such as the currently pre-eminent "Google" respond rapidly and accurately in non-rigorous, but demanding, civilian situations. The military analyst, while retaining the same needs for excellent interface, speed, accuracy, relevancy and scope, also requires a greater assurance that data ascertained represents sufficient, and accurate results of relevant materials. The high performance computing centers provide a common ground where these diverse database professionals meet. Synthesizing the advances from all of these disparate fields arguably provides the synergy necessary to meet the rapidly expanding needs of the FMS community.

Data Mining

Data mining techniques are defined here as the extraction of useful patterns and modes from data sets that are often large. More particularly we, and others, use the term to specially imply the extraction of insights from databases for which that data structure was not originally designed.

Some authors have described data mining as lying at the intersection of statistics, machine learning, data management, pattern recognition, artificial intelligence and other related disciplines. The authors see it as the application of myriad techniques to accomplish its goals, but not subsuming all of these techniques into itself. Its focus on "…unsuspected relationships ..." and summarizing data in "… novel ways that are both understandable and useful …" (Hand, 2002) is the capability that is seen as most promising for FMS data analysis.

Data mining can be more generally said to require some significant effort in each of the following tasks:

1. initial data analysis to gain understanding of organization and visualization possibilities
2. an attempt at describing a loose-fitting, but acceptable model of the data under analysis
3. the creation of a model capable of predicting the results and the relationship of those results to certain input parameters
4. the final analysis of the data sets with the final product being not only the discovered relationships, but also the real-world insights that such relationships support

Simulations of the order discussed in this paper may generate as many as 1,600,000,000 data points (1,600 data elements per entity for 1M entities.) With such vast amounts of data, not all useful analysis can be done real-time, nor is it optimally productive to do so. The common result is that reams of recorded data sets are discarded as too cumbersome to be of analytical use. Data mining tools offer promise in that they allow the analyst to find useful information and patterns amidst the mass of data points even after the simulation is completed

With the power of scalable parallel processor supercomputers, once simulation results have been characterized and values ascribed to various outcomes, the recursive analysis of the data will undoubtedly find useful new views of that which critical to the outcome. For example, using data from numerous iterations of a flight simulation designed solely for training, one might find a pattern of inexperienced pilots tending to overshoot their targets. Without making the effort to analyze such data or to create effective tools for sifting through the vast amount of noise to find useful information, the opportunity to discover such useful patterns is lost. Data mining tools help to isolate not just the story from the activity, but the wisdom to be gained there from.

The data mining process does require efforts beyond traditional simulation analysis. Normally data mining requires all or some of the following:

1. Achieving a thorough understanding of the representations' inherent characteristics and organization (*e.g*., parameters of the entities, descriptions of their activities).
2. Selecting methods of defining and comparing the data in such a way that it will yield quantifiable results that can be compared (*e.g*., losses, mission success, time).
3. Discovering, defining and applying an algorithm to compare results with input parameters (multi-variate studies of data sets).
4. Analyzing and implementing those data management techniques that will enable and facilitate steps one through three.

The tasks above need not make demands on the structure of the data nor the means for attaining it. By its very nature, data mining presents low cost opportunities for gains in insight and understanding from simulations of almost any sort with little to no impact on or cost to the simulation itself.

Data mining has historically proven to be an effective tool in numerous fields. Trigon Blue Cross Blue Shield uses data mining techniques to identify early indicators of serious disease, thus allowing them to effectively treat patient before they become seriously ill. Data mining methods helped retailer Williams-Sonoma save millions in advertising costs without losses in sales by creating a targeted system for catalog distribution. Using data mining technologies, banks have developed better credit scoring models that more accurately predict applicants that may default on loans. In science, data mining techniques have been used to identify new binary stars by using radio telescope data collected for mapping, but which serendipitously contain the characteristic oscillation frequencies of such stars yet undiscovered (Moore, 1998). Similarly, the authors believe that unsuspected insights that will save lives, money and missions lie deeply imbedded in the data being generated today by FMS.

As increasingly complex simulations produce larger and larger datasets, data mining techniques will help the analyst sift through that mountain of data in order to find and quantify useful relationships and patterns. Increased computing power and faster computational capabilities only increase the opportunity to find useful patterns. While the authors do not represent that data mining will solve all problems nor discover all relationships of interest, they do accept the notion that it has the potential of discovering many new relationships, some of which may enable significant new capabilities or prevent monumental losses.

Evolutionary Computing

Another area of significant opportunity lies in the application of the techniques described by the Fogels in their work on Evolutionary Computation. (Fogel, 2000) Many of the new battlefield challenges represented by the relationships of the data described above are far removed from the current understanding of defense strategies. They will not be observed, presumed or described by even the most rigorous analysis of the data. Novel and asymmetric threats are continually and rapidly evolving. These new threats are being driven by groups whose one remaining effective weapon may be their tactical innovation and the resultant element of surprise. In this they are aided and abetted by their remoteness from the defense analysts in topics such as their value system, goals, training, and *zeitgeist*.

The family Fogel presents a way to examine a virtually unlimited horizon of possibilities by using techniques that perturb the physically accurate simulations of the world without regard to the constraints of the expectation or creativity. They replace the rule-based foundation of the Monte Carlo simulations with the concept of an entity that is able to freely roam the range of possibilities, with an appropriate feedback loop to help in optimizing the path to the goal. Basing their work on the areas of artificial intelligence, expert systems and neural net training, the evolutionary computer scientists further look to the biological paradigms popularized by Charles Darwin in his work on the evolution of animals. This group eschews slavish imposition of genetic rules and prefers to let electronic intelligence finds its own path in parallel with biologic evolution.

Applying the concepts laid out by these evolutionary computational scientists has the promise of establishing unimagined methods and threats. Should the evolutionary computing process result in the identification of an unnoticed vulnerability or the determination of a new threat, the defenses could be altered, steps could be taken to ameliorate the losses, or contravening punitive actions aimed at the attackers could be imposed.

Genetic Algorithms

In a variant of the work by the Fogels, David Goldberg reports significant success in applying more stringently biologic rules to his analysis (Goldberg, 2002). He sees the genetic evolutionary driver as having been tested over the millennia and therefore not likely to be deficient. His application of genetic rules is similarly successful in the test phases of his work. He feels the insights he gains are more likely to be in accord with the behaviors observed in actual life. Dr. Goldberg has used his techniques to model both organizational entities such as small populations and physical phenomena such as gas pipelines. His approach does suggest a very supportable relationship between his data and the observed data in the population under study and the pipeline under observation.

The selection between evolutionary computing and genetic algorithms can be left to the user as an exercise. In each case, the identification of a novel concept would have to sustain the challenge of reason and the governmental vetting process prior to funding a new defense or the acceptance of a new approach. The *caveat* to be remembered is not to disregard novel approaches and valid insights.

Monte Carlo Analyses

Many of the simulations in use by the services today rely heavily upon Monte Carlo techniques. These simulations have a pre-established rule set and distribution or likelihood for each major activity as was described above. As noted earlier, these simulations are not deterministic and often the same basic initial definition is executed several times (hundreds of runs are not uncommon) to examine the distribution of the final outcomes, (Horne, 1999). This work is often analyzed by plotting out a series of two dimensional solution spaces on a three dimensional graph, as in Figure 2, and visually identifying the optima and their relation to one another for each pair and then estimating the interrelation of the multivariate group.

Based on the work of a physicist at Caltech, the OTCI company has developed a tool that can quantify the degree to which the input parameters affect the final outcome. This can be done in *n* dimensions, which would be an improvement on the visual analytical procedure outlined above. Further, this procedure

yields very interesting results in fewer runs, sometimes orders of magnitude fewer (Johnson, 1999). The technology is currently implemented for financial analyses, but could be "ported" over to battlefield simulation analyses with a high expectation of efficacy and a reasonable hope for better analytical products.

.

IMPLEMENTATION EXAMPLES

There are both examples of successes and of well-documented plans for how the techniques reviewed above can be implemented in the future, including three easily envisioned ways to approach the scalable parallelization of a simulation. First, design the code as a well-parallelized program from the beginning. Second, after reviewing existing code, completely rewrite an existing code base in a scalable parallel manner. Third, take the code as it is and implement a new "wrapper" around the code that makes it scalable. Two of the noted implementations have been seen in the intelligent agent, non-deterministic variety of simulations: SF Express/JESPP (Joint Experimentation on Scalable Parallel Processors) and Project Albert.

In SF Express, the ModSAF code was enhanced with communications routers written in Message Passing Interface (MPI). The routers enabled scalability both within the SPP mesh and across the nation. This was a successful example of distributed, heterogeneous supercomputing. Scalability was measured in comparing the times experienced in communications activity as the size of the sample increased.

In JESPP, the team was asked to make the JFCOM's JSAF simulation much more scalable (see Figure 5) and portable to Linux clusters of the 256 node class. JFCOM needed to "field" more than a million vehicles in an urban setting.

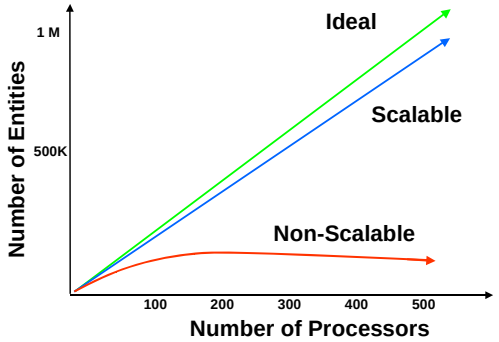


Figure 5. *Notional Scalability using Mesh Routers*

This was necessary to test many concepts, including the need to assess various simulated sensor platforms and associated systems in their ability to discriminate combatants from non-combatants.

Previous implementations on LANs did not simulate more than 30K vehicles. In making the code more scalable and running it on a series of Linux clusters, the JESPP team was able to achieve more than 1,000,000 vehicles (Lucas, 2003 and Helfinstine, 2003). The approach they used is worth future consideration. It entailed the careful study of the simulation code in use, JSAF, and then constructing a system of very scalable software routers to make the code capable of effectively using the hundreds of processors in large Linux clusters. The code base itself was not significantly impacted and the software routers were designed to accommodate as best they could the almost daily changes in JFCOM's growing and dynamic needs, which caused frequent modifications to the underlying JSAF code. To achieve scalability, the effort engaged computational scientists with extensive backgrounds in physical science simulations who had the intellectual and creative skills to rapidly understand and to effectively enhance the code.

Project Albert has taken a different track. From the beginning, the code was constructed with a kind of built-in scalability. The basic idea of Brandstein and Horne was that Albert would not have a fully fleshed out simulation, but would be convey the "essence" of the activity running in a very small module that can be run over and over. The Project Albert crew has worked very closely with the parallel-computing experts at the Maui High Performance Computing Center. The code base is kept quite small by design. It has less than ten per cent of the lines of the JSAF code.

While this *de novo* approach has the benefits of elegance and simplicity, it is, by definition, only open to the developers who are producing entirely new programs. The authors find that the FMS community frequently adapts and expands existing code and one recent major new, "bottoms-up" FMS program was recently terminated (Tiron, 2003). Nonetheless, developers of new systems should resist being seduced by the ease of single-processor designs, as experience has now shown that there will be pressure to expand along the dimensions of complexity, resolution, and magnitude, hence requiring or benefiting from parallel processing.

In looking at appropriate data analysis techniques to implement, one issue that must be settled is that of the users' goals in this analytical process. As this seems rather intuitive, there is an inclination to skip, or at least slight, this step. Previous scholarship bears

careful attention. Going outside the narrow confines of FMS, it is clear that Operations Researchers (OR) have studied comparable issues for a century. Some members of the OR community have been active in helping simulation groups understand their goals and analyze their results (Sanchez, 2002).

Ongoing Work

Many of these programs are ongoing. JFCOM has committed to the JESPP project for the foreseeable future. The award of the Distributed Center cluster to JFCOM will provide a natural home for the technical life of the cluster. Additional clusters can be enlisted to provide more processing power, and the expandable capabilities of the JESPP scalable routers produce the ability for the JSAF code to utilize all of the processors that can be gathered for an exercise. An extension of this work into the analysis of homeland security issues for air traffic control has been advanced and the authors consider this a well-founded application of the concept of taking code as is and enhancing it with augmenting wrappers.

The Albert project continues to be a vital part of the study of maneuver warfare. In addition to studying new ways of utilizing large parallel computers, Albert seeks out new ways to analyze the huge amounts of data presented by the multiple run method. As the project increases the number of important variables, the difficulty in visualizing multi-dimensional solution spaces may find resolution in the work of Monte Carlo simulators in the financial community.

This generates an analytically valuable data surfeit that may be found in the vast quantities of data that could be collected from training simulations. As pilots, sailors and tankers train in their simulators, their activities' data make a fertile field for other analyses.

Adopting either of the two methods discussed above, designing for scalability or augmenting with scalable wrappers, should produce several benefits. First, both should create not only early scalability but imbue the code with an ability to scale further and make early use of new processor technologies. Second, experience has shown that the parallelization process itself frequently improves the serial code and, not infrequently, leads to insights into the subject phenomena. Third, careful application of these techniques should not disturb the development or use of the delivered code. Fourth, cost, schedule and performance can be kept in balance.

There are actions that will reduce potential disruptions and produce the best results. The most important of these may seem obvious, but it is not infrequently overlooked - the reliance on experienced parallel computational scientists. Parallel programming is a unique skill-set. Attempts to automate the process of parallelizing code have not been particularly fruitful, especially in programs where coarse-grained parallelism is appropriate. A research and development group seeking to make their code scalable would be well advised to identify a successful effort implementing comparable code on an SPP and then engage the parallel programmers who were responsible and who have exhibited a transferable aptitude.

TECHNOLOGY IMPACT

It is the firm conviction of the authors that the technology detailed above will prove to be a vital asset for the FMS community and then have an essential impact on the defense of the nation. The necessity of dealing with the commingling of combatants and non-combatants, the current mandates to conduct operations with minimal disruption of civilian infrastructure, and the ability to wage effective warfare against an asynchronous enemy all will be addressed more completely using the advanced techniques discussed.

However, the FMS community has not shown as much acceptance of these technologies as might have been expected. At the 2003 IITSEC meetings, only the JFCOM papers on the use of Linux Clusters evidenced implementations in everyday use (Lucas, 2003; Helfinstine, 2003; and Williams, 2003). The three other papers mentioning these topics (Pratt, 2003; Schiavone, 2003; and Mielke, 2003) took valid, but much more theoretical, perspectives. This year may not show much of an increase. A review of the submitted titles for IITSEC 2004 reveals the lack of a single mention of the terms Beowulf, supercomputer, parallel processing, data mining, evolutionary computing, sensitivity analysis or high performance, although these issues are discussed in other papers by our JESPP team members who were instrumental in the effort to enable larger-scale entity counts in the Urban Resolve experiments at JFCOM.

The ability of the analyst to distinguish between non-combatants and enemy forces hiding among them relies on increasingly effective sensors, well-designed analytical systems, and advanced training in realistic environments. Current limitations in resolution, entity count and sophistication of behavior interfere with all of these. Simulation experimenters report that analysts engaged in early exercises had so few civilian entities

in their environment that they were inclined to opt for destruction of all vehicles under observation, when there was doubt as to their identity. Reasonable choices were also restricted when the number of civilians was smaller than the number of enemy combatants, a condition driven by the lack of compute capacity on platforms consisting of PCs on LANs.

The lack of sophistication also can render an exercise less meaningful. Human operators are very sensitive to behavior differences. If computer constraints enforce very simplistic behaviors on modeled civilian vehicles, the operators quickly can distinguish them from the more complex behavior capabilities of the combatant vehicles, *e.g*. if the simulation controllers turn off collision avoidance to save on inter-visibility calculations, the operators will quickly perceive that any vehicle that passes right through another is not a combatant. Neither good training nor good analytical input can result from similarly constrained conditions.

Additionally, the not uncommon reliance on SME reviews of simulations, while effective and useful, may be missing valuable insights. These insights might otherwise lead to new strategic concepts or prevent overlooking significant vulnerabilities. Not yet having faced the unknown enemy of the future, not knowing its mind-set, and not having the luxury of learning at a leisurely pace, the simulation community would be well-advised to take advantage of the expanded capabilities presented above in the section on advanced data analysis techniques.

Orderly retrieval of information using the latest database techniques will assist human analysts in pursuing intuitive leads. The innovative techniques representing data mining can be invoked to extract even more esoteric concepts and bring these to the attention of the analysts for confirmation and analysis. This gives real hope for identifying asymmetric tactics that might not be foretold by traditional military analysis. The concepts of evolutionary computation, genetic algorithms, and Monte Carlo sensitivity analyses also show promise in making sure nothing is missed in the search for security.

A Development Path: Successful Rapid Prototyping

Transitions from current simulation methods to full exploitation of present and near-term computational capabilities and practices take effort and significant experimentation. It is perhaps best to illustrate the process with a particular example: a suite of large field-of-view sensors attempting to detect isolated "suspicious" behaviors within a large population of normal (i.e., "civilian") entities. This has been, in fact, a major thrust of ongoing JSAF developments, with the Simulation of the Locations and Attack of Mobile Enemy Missiles (SLAMEM™) surveillance/tracking software system fed by detections from simulated civilian vehicles (euphemistically called "clutter") within the JSAF simulation

.

The JSAF/SLAMEM combination has so far been rather fruitful. For present purposes, it is sufficient to consider three particular items:

1. In order to support large numbers of clutter entities, the clutter models within JSAF had to be quite simplistic, with, *e.g*., very little "self-awareness" among clutter entities.
2. While the SLAMEM-JSAF system exploits a number of clever procedures to distribute much of the computational burden (in particular, some of the simulated signal processing), the tracking and situation-assessment procedures within SLAMEM were originally done on a single processor, thus providing a significant constraint on the size of the underlying simulated scenario.
3. The very large numbers of simulated detections within a typical SLAMEM-JSAF were largely "unexploited", beyond the immediate task of driving track formation and feeding operator displays.

There are a number of straightforward technology "patches" for many of these problems, including parallelized tracking algorithms, and a much richer, distributed database system supporting data mining and "discovery" activities. Incremental developments along these paths are inevitable. The problem, of course, comes with the word "incremental." The standard practice of inserting pieces of computational technology, as though one were simply using higher clock rate processors, drives the system along a path dictated by "ease of insertion" rather than ultimate end-user needs. In the authors' opinion, it is definitely progress, but it is unlikely to be progress that will ever catch up to available capabilities.

Consider, again, the conceptual SLAMEM problem. At a high-enough level, the outcome of present experimentation must point to the desired or idealized product: Operators are watching displays of highly processed tracking results, looking for indications of both "suspicious behavior" and reactions to interdiction activities. Human interpretation of these data will always be subjective. The details that could be provided by scaled computational power alone are

overwhelming, if not overwhelmingly useless. Operators needed dynamic access to the available data at several scales of both "resolution" and "historicity" in order to assign likelihoods to the important bottom-line issues of asymmetrical combat.

It can be argued that the ongoing FMS development path would not reach this goal (or rather, if it does, it will do so very, very slowly). This is not to say that standard practices should be abandoned! The authors maintain that incremental development and implementation is the only sane way of improving the state of the art while maintaining capabilities.

The authors suggest a parallel development track is needed, emphasizing the "top-down" approach with the goals of identifying: 1) inherent limitations in standard practices and 2) technology needed to resolve the identified problems. The intent of the second point must be clarified/emphasized. Rather than ask the implicit "standard practices" question ("What incremental capabilities can be added through readily available technology?"), a very different question must be asked for optimal implementation: "What technology is needed to achieve required capabilities?" Viewed from the perspective of the idealized ultimate user, the system to which JSAF/SLAMEM activities are pointing must be database driven and must address the following questions: "What information will best aid the decision maker?" and "What automated discovering and mining procedures are needed to make this data perceivable to the decision maker?"

CONCLUSION

There are several new technologies that are available and are of demonstrated utility to the simulation community. There also exists a body of practice that makes adoption of these capabilities more productive and less disruptive.

September 11, 2001, gave a new immediacy to the task of adequately preparing for unexpected threats. While the techniques proposed in this paper may not have helped avert that tragedy, the authors maintain those techniques may well increase the opportunity for the analysts to discover future threats and assist in working out the best way to defend against such destruction. Considering the huge losses that the nation incurred from that one attack, the efforts required in implementing the described techniques pale in comparison.

Data interpretation is a critical task in any war, including the war on terrorism. Simulation systems may well benefit from enhanced data interpretation and that should do much to provide a real-time laboratory for refining and exploiting advances in data analysis that have been made over the last decade.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the members of the ISI JESPP team who have contributed to this paper through their efforts and their intellectual stimulation. Much of the success reported here came from the Joint Experimentation on Scalable Parallel Processor project, initiated, directed and funded by the Joint Forces Command and to a very large degree conducted on the compute assets of the Maui High performance Computing Center and other members of the High Performance Computing Modernization Program. Without the support and encouragement from all of the above, none of this would have been possible. This material is based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Amdahl, G.M. (1967). Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings*, 30, 483-485.

Ben-Ari, E. (1998). Mastering Soldiers: Conflict, Emotions and the Enemy in an Israeli Military Unit. *New Directions in Anthropology*, 10. Oxford: Berghahn Books.

Brandstein, A. (1998). Data Farming: A Meta-technique for Research in the 21st Century. *Maneuver Warfare Science* Quantico, VA: United States Marine Corps Combat Develop Command, 93-99.

Brunett, S., Davis, D., Gottschalk, T., & Messina, P. (1998) Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments, *The Seventh Heterogeneous Computing Workshop*, Orlando, FL.

Cebrowski, A., & Garstka, J. (1998) Network Centric Warfare: Its Origin and Future, *Naval Institute Proceedings*, 124(1), 28-35.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J. (2002) Reflections on Building the Joint Experimental Federation. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL.

Ceranowicz, A., Dehncke, R.& Cerri, T. (2002) Moving toward a Distributed Continuous Experimentation Environment, *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL.

Dubik, J. (2003) Comments made by Major General James M. Dubik, J9, US Joint Forces Command, at I/ITSEC in Orlando, FL.

Foster, I. & Kesselman C. (1997) Globus: A Metacomputing Infrastructure Toolkit. *Intl J. Supercomputer Applications*, 11(2), 115 –128.

Fogel, D. (1995) *Evolutionary Computation*. New York: IEEE Press.

Gibson, D. (2003) Casualty Estimation in Modern Warfare. *Army Logistician*, Nov-Dec 2003 35(6), 34. Fort Lee, Virginia.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Boston, Massachusetts: Addison-Wesley Publishing Co.

Garlan, D. & Shaw. M. (1993) *An Intro to Software Architecture: Advances in Software and Knowledge Engineering*. World Scientific Publishing, I.

Graebener, R., Rafuse, G., Miller, R. & Yao, K-T. (2003) The Road to Successful Joint Experimentation Starts at the Data Collection Trail. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL.

Hand, D., Mannila, H., Smyth,. P. (2001). *Principles of Data Mining (Adaptive Computation and Machine Learning).* Boston, MA: MIT Press.

Harnad, S. (1989) Minds, Machines and Searle. *Journal of Theoretical and Experimental Artificial Intelligence*.1, 5-25.

Helfinstine, W. Torpey, M. & Wagenbreth, G. (2003) Experimental Interest Management Architecture for DCEE. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Hill, R. W., Gratch, J., & Rosenbloom, P.S. (2000). Flexible Group Behavior; Virtual Commanders for Synthetic Battlespaces. *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Spain.

HPCMP (2004) Computational Technology Areas. Forces Modeling and Simulation CTA, (Online)May 13, 2004, http://www.hpcmo.hpc.mil/Htdocs/CHSSI/cta_description.html

Horne, G. (1999) Maneuver Warfare Distillations: Essence not Verisimilitude. *Proceedings of the 1999 Winter Simulation Conference. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans,* Phoenix, AZ

Hull, J. (2002). *Options, Futures and Other Depravities, 5th Ed.* New York: Prentice Hall.

Johnson, E., Bergman, L. & Spencer, B. (1999). Intelligent Monte Carlo Simulation and Discrepancy Sensitivity. In P.D. Spanos (ed.), *Computational Stochastic Mechanics (pp. 31-39).* Balkema, Rotterdam.

Kevrekidis, I., Gear, C., Hyman, J., Kevrekidis, P., Runborg, O. and Theodoropoulos, D. (2003) Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*. 1, 715-762.

Kleijnen, J., Sanchez, S., Lucas, T., & Cioppa, T., in print, A User's Guide to the Brave New World of Designing Simulation Experiments, under revision for *INFORMS Journal on Computing*.

Krishnaprasad, S. (2001) Uses and Abuses of Amdahl's Law. *The Journal of Computing in Small Colleges* 17(), 288–293.

Litzkow, M., Livny, M. & Mutka, M. (1988) Condor - A Hunter of Idle Workstations. *Proceedings of the 8th International Conference of Distributed Computing Systems*, 104-111.

Lucas, R. & Davis, D. (2003) Joint Experimentation in Scalable Parallel Processors. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Marshall, S.L.A. (2000). *Men Against Fire: The Problem of Battle Command*, Norman, Oklahoma: University of Oklahoma Press.

Messina, P. C., Brunett, S., Davis, D. M., Gottschalk, T. D. (1997, April) Distributed Interactive Simulation for Synthetic Forces, In J. Antonio, (Chair), *Mapping and Scheduling Systems, International Parallel Processing Symposium*, Geneva, Switzerland.

Mielke, R. & Phillips, M. (2003) Development and Application of an Academic Battle Lab. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL

Moore, G. (1965) Cramming more components onto integrated circuits. *Electronics* 38 (8), 114-117

Moore, R., Prince, T., Ellisman, M. *(1998)* Data-Intensive Computing and Digital Libraries. *Communications of the ACM*.

Pratt, D. & Henningger, A. (2003) Load Balancing for Distributed Battlefield Simulations: Initial Results. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*. Orlando, FL

Sanchez, S.M. and Lucas, T.W. (2002) Agent-based Simulations: Simple Models, Complex Analyses, Invited paper. *Proceedings of the 2002 Winter Simulation Conference*. Institute of Electrical and Electronic Engineers: Piscataway, NJ.

Sanne, J. (1999). *Creating Safety in Air Traffic Control*. Unpublished doctoral dissertation, Institute of Tema Research, Linköping University, S-581 83 Linköping, Sweden.

Schiavone, G. Dolezal, M., Tracy, J., Secretan, J., & Mangold, L. (2003) Beowulf Supercomputing for Mobile Applications. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*. Orlando, FL

Sterling, T., Salmon, J., Becker, D. & Savarrese, D. (1999) *How to Build a Beowulf.* Boston, MA: MIT Press.

Tiron, R. (2003), Pentagon Cancels Program with Checkered Past, *National Defense Magazine*,. From http://www.nationaldefensemagazine.org/article.cfm ?Id=1081

USNews. (1988, August 15) *Was stress the villain in the air tragedy?* U.S. News and World Report, 105(7), 13

Van der Steen, A., & Dongarra, J. (2003) Overview of Recent Supercomputers. *Top 500 Supercomputer Sites*. (Online) February 12, 2003, http://www.top500.org/ORSC/2003/

van Lent, M. & Laird, K. (1998). Learning by Observation in a Complex Domain. *Proceedings of the Knowledge Acquisition Workshop*. Banff, Canada.

Appendix G

Advanced Message Routing for Scalable Distributed Simulations

Brian Barrett
University of Southern California
Marina del Rey, CA
bbarrett@isi.edu

Thomas Gottschalk
California Institute of Technology
Pasadena, CA
tdg@cacr.caltech.edu

ABSTRACT

The Joint Forces Command (JFCOM) Experimentation Directorate (J9)'s recent Joint Urban Operations (JUO) experiments have demonstrated the viability of Forces Modeling and Simulation in a distributed environment. The JSAF application suite, combined with the RTI-s communications system, provides the ability to run distributed simulations with sites located across the United States, from Norfolk, Virginia to Maui, Hawaii. Interest-aware routers are essential for communications in the large, distributed environments, and the current RTI-s framework provides such routers connected in a straightforward tree topology. This approach is successful for small to medium sized simulations, but faces a number of significant limitations for very large simulations over high-latency, wide area networks. In particular, traffic is forced through a single site, drastically increasing distances messages must travel to sites not near the top of the tree. Aggregate bandwidth is limited to the bandwidth of the site hosting the top router, and failures in the upper levels of the router tree can result in widespread communications losses throughout the system.

To resolve these issues, this work extends the RTI-s software router infrastructure to accommodate more sophisticated, general router topologies, including both the existing tree framework and a new generalization of the fully connected mesh topologies used in the SF Express ModSAF simulations of 100K fully interacting vehicles. The new software router objects incorporate the scalable features of the SF Express design, while optionally using low-level RTI-s objects to perform actual site-to-site communications. The (substantial) limitations of the original mesh router formalism have been eliminated, allowing fully dynamic operations. The mesh topology capabilities allow aggregate bandwidth and site-to-site latencies to match actual network performance. The heavy resource load at the root node can now be distributed across routers at the participating sites.

ABOUT THE AUTHORS

Brian Barrett is a programmer analyst on the JESPP project, Information Sciences Institute, University of Southern California. Brian's research has focused on communication issues for large-scale high performance computing systems. While at Indiana University, Brian was a lead developer of the LAM/MPI implementation of the Message Passing Interface (MPI) standard. He received a B.S. from the University of Notre Dame and an M.S. from Indiana University, both in Computer Science.

Thomas D. Gottschalk is a Member of the Professional Staff, Center for Advanced Computing Research (CACR) and Lecturer in Physics at the California Institute of Technology. He has been with CACR for nearly a decade. Much of his research has been on the use of parallel computers to simulate various physical phenomena. His instructional duties include his upper division course on Statistics for Physics Graduate students. He received a B.S. in Physics from Michigan State University and a Ph.D. in Theoretical Physics from the University of Wisconsin.

Large Scale Forces Modeling and Simulation

Recent experiments within the Joint Forces Command (JFCOM) Experimentation Directorate (J9) demonstrate the feasibility of forces modeling and simulation applications in a large field of play with fine-grained resolution. Simulating such battle spaces requires large computational resources, often distributed across multiple sites. The ongoing Joint Urban Operations (JUO) experiment utilize the JSAF application suite and the RTI-s Run Time Infrastructure to scale to over 300 federates distributed across the continental United States and Hawaii (Ceranowicz, 2002). The JUO exercise has shown the scalability of the JSAF/RTI-s infrastructure and of interest-based, router-managed communication. At the same time, the simulation has highlighted a need for improvements in the communication architecture.

Figure 1: Software routing topology for the JUO exercise.

The current JUO network topology is a tree of software routers (see Figure 1 for wide area network diagram). The hub and spoke network model introduced by this tree infrastructure increases latency between distributed sites and exposes the entire network to a single point of failure. The tree topology also poses a scalability limitation within the distributed sites. It is our belief that an improved routing infrastructure is required for the continued success of large-scale entity level simulations, particularly as entity counts and complexity/fidelity increase.

This paper presents an improved routing architecture for large-scale HLA environments, using fully connected meshes as the basic topology. These mesh routers provide a scalable solution for interest-managed communication, as well as a more accurate mapping of software routing to available network topologies.

Scalable Parallel Processors

The JUO exercise requires a computational ability unavailable using traditional groups of workstations. Scalable Parallel Processors (SPPs) provide the required computational power, with modest increase in development and execution effort (Lucas, 2003). A SPP is a large collection of processing elements (nodes) connected by a fast communication network. Common SPPs include the IBM SP, SGI Origin, Cray X1, and Linux clusters. Traditionally, SPPs provide services not available in a group of workstations: high speed networks, massive disk arrays shared across the entire resource, and large per-CPU physical memory. In addition, SPPs generally have uniform environments across the entire machine and tools for scalable interactive control (starting processes across 100 nodes takes the same amount of time as it does across 10).

Linux clusters have recently become a suitable platform for the high performance computing community and are therefore readily available at Department of Defense Major Shared Resource Centers. These clusters are ideal platforms for use in the JUO exercise because of their close heritage to the Linux workstations used in the interactive test bays. Although there is additional

software to tie the cluster into one SPP, the basic libraries, compiler, and kernel are often the same on a cluster as on a workstation.

RTI-s

RTI-s provides the HLA Run Time Infrastructure (RTI) for the JUO federation. RTI-s was originally developed for the STOW exercises, to overcome the scalability and performance limitations found in RTI implementations at the time. It should be noted that RTI-s is not a fully compliant HLA/RTI implementation. Specifically, it does not implement timestamp ordered receives, ownership transfer, and MOM interactions. In addition, federates discover new objects at first update, rather than at creation time. The JSAF applications are receive-ordered by design and are optimized to respond best to delayed object discovery, so these limitations are not constraining in the existing environment.

RTI-s utilizes a flexible data path framework (an example of which is shown in Figure 2), which allows for use over a number of communication infrastructures. Currently, there is support for multicast UDP, point-to-point UDP, point-to-point TCP, and MPI (using a send/receive architecture). Bundling and fragmenting of messages is provided by components that can be reused for TCP and UDP communication. Kerberos authentication for data packets has been implemented for TCP communication.

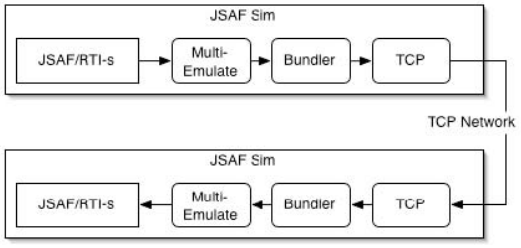


Figure 2: RTI-s data path architecture for TCP communication.

Point-to-point modes in RTI-s uses separate routing processes for communication. The routers provide data distribution and interest management for the federation, which would be too heavy for a simulator to handle. Presently, a tree topology (Figure 3) is used for connecting routers. A tree presents a simple structure for preventing message loops, as there are no potential loops in the system.

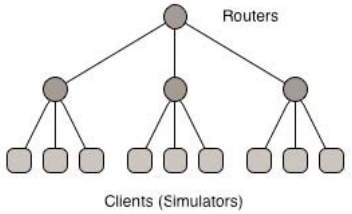


Figure 3: Tree topology used by RTI-s for point-to-point message traffic.

Synthetic Forces Express

The Synthetic Forces Express (SF Express) (Brunett & Gottschalk, 1998) project first demonstrated the suitability of both the SPP and mesh router concepts for discrete entity modeling. The SF Express project extended the ModSAF simulation engine (Calder, 1993), focusing on the communication protocols to extend scalability.

In December 1996, the SF Express team achieved a 10,000 vehicle simulation using a single 1,024-node Intel Paragon machine. Message routing within the SPP used the Message Passing Interface (MPI) (MPI Forum, 1993). Later work allowed the code to run on multiple SPP installations across a variety of networks by introducing gateways between SPPs. The gateway routers were connected using UDP. With these improvements, the project achieved a simulation of 50,000 vehicles using 1,904 processors over six SPPs.

The structure of the SF Express router network is shown in Figure 4. The basic building block for this architecture is the triad shown on the left, with a "Primary" router servicing some numbers of client simulators. Two additional routers (known as the “PopUp” and “PullDown” routers) complete the basic triad. These routers distribute (PopUp) and collect (PullDown) messages from client simulators outside the Primary’s client set. The SF Express architecture scales to increased problem size by replicating the basic triad and adding full up down communication links among the triads, as shown in the right hand side of Figure 4.

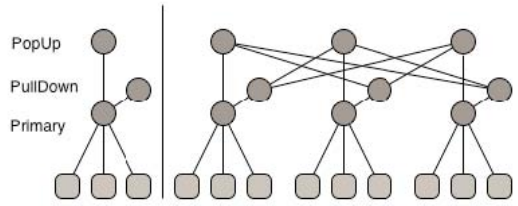


Figure 4: Basic building block of the SF Express routing network (left) and an example mesh topology (right).

While the SF Express project was quite successful, it had no life beyond a number of 50K-100K entity simulation demonstrations. This was expected, for a number of reasons. For example, the algorithms and software developed for that project were not compatible with ongoing SAF developments (e.g., the move to RTI). Finally, the MPI-based communications used within the SPPs did not tolerate the restarts and process failures found during a long running exercise

Designing for Scalability

As previously mentioned, the JSAF/RTI-s application suite currently scales to over 300 federates and over a million entities (including simple clutter). However, current routing topologies limit the scalability of the overall system. In order for an interest-based communication infrastructure to scale, three conditions must hold over an arbitrary interval of simulation time:

- A given client must generate a bounded number of messages
- A given client must receive a bounded number of messages.
- Given the previous two points, the communication through any given router must also be bounded

An interest management system and careful federate design achieve bounded client communication. Bounded router communication is a function of network design and can be achieved using a mesh topology.

Interest Management

The aggregate amount of data produced by the JUO federation is greater than any one federate is capable of processing. An interest management system is used to limit the amount of data a federate must process (Rak, 1997). The federate declares which information it is interested in ("e.g., red force tanks in position cell X") and the RTI is responsible for ensuring only this subscribed information is received by the federate.

When used in a multicast environment, RTI-s utilizes the concept of multicast channels for filtering, with interest states having associated channels. The message is multicast to the federation's network and filtered on the receiving side. The receiver filters the message at the kernel level, so the application never sees messages for interest states it is not interested in. Overhead when no interest states are set is relatively small, but non-zero. Due to the limited number of available multicast channels, the number of interest states is limited (increasing the amount of traffic associated with each interest state).

When running in point-to-point mode (using either TCP or UDP), interest management is send-side squelched. Software routers maintain interest state vectors for each connection and only send messages to clients that have expressed interest in a message type. The overhead for a federate to exist in the federation without any expressed interest is almost zero. Because interest states are not tied to hardware and operating system limitations, the number of available interest states is bounded only by how much memory can be allocated to interest vectors. This is an enormous improvement over multicast IP. It was also one of the innovations of SF Express.

An interest management system provides only the infrastructure for bounding the data flowing out of and into a particular simulator. The simulator must show care in declared interest states to prevent subscribing to more data than it is capable of processing. For the purposes of analyzing the scalability of routing infrastructures, we assume that the simulator limits interest declarations to guarantee bounded communication. In both the earlier SF Express and current JUO experiments, this assumption appears valid.

Routing Scalability

The scalability of the basic Mesh Router network is easily argued as follows. It is first necessary to assume that the underlying simulation problem itself has a scalable solution. This means a

bounded message rate on the Primary PopUp and PullDown Primary links within a basic triad, and bounded Up Down message rates within the interconnection links of the full network. The impediments to complete scalability of the mesh architecture have to do with interest declarations among the upper router layers. Each PullDown must announce its interest to every PopUp. In principle, these interest broadcasts could be made scalable through an additional network of communication nodes (at the associated cost of increased latencies for interest updates). In practice, however, these interest updates were not frequent enough to cause any difficulties in SF Express simulations with as many as thirty triads in the full mesh. An experiment with a similar setup using the current infrastructure shows similar results. This formally non-scaling component is, in fact, a sufficiently tiny component of the overall communications load that implementation of the "formal" scalability cure is not warranted for present or near-term simulation scenarios.

Routing Flexibility

The scalability issues with the tree router topology of RTI-s have been discussed previously. Tree topologies also map poorly onto physical wide-area networks. Figure 1 shows the route taken for any message crossing multiple sites in the JUO exercise. The path taken for a message to go from Maui to San Diego is sub-optimal: the data must first travel to Norfolk, then back to the west coast. This extra transmission time increases the latency of the system, which lowers overall performance. Since wide-area links often have less bandwidth available than local area networks, such routing also places a burden on the Virginia network infrastructure, which must have bandwidth available for both the incoming and outgoing message in our Maui to San Diego example.

Figure 5: Advanced routing topology for JUO exercises.

The mesh routing infrastructure provides a better utilization of physical networks by sending directly from one source to destination router. The network infrastructure is free to route messages in the most efficient way available. Figure 5 shows one possible routing topology for the JUO exercises, using mesh routers to minimize the distance messages must travel.

In an ideal world, the entire federation would use one fully connected mesh for message routing. The actual routing of messages would be left to the physical network infrastructure, which has over 30 years experience in optimizing data. However, such a configuration is often not feasible due to performance or protocol availability. Local area communication is usually over TCP, pushing error detection from RTI-s to the network stack. Over wide area networks, however, TCP suffers bandwidth degradation proportional to latency, so UDP is used for these connections. Some SPPs provide neither TCP nor UDP on computer nodes, instead providing MPI over a high-speed network) or provide public access only on a small subset of the machine. Given these restrictions, a fully connected mesh is often not a feasible design.

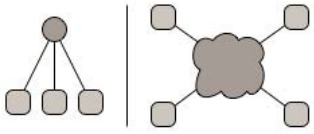


Figure 6: The basic building blocks for a Mesh Router topology: tree (left) and mesh (right).

The mesh router provides the ability to design a flexible network topology that meets the

constraints of the network infrastructure while providing the ability to design a scalable system. The mesh router's topology is constructed by combining two building blocks: a tree (Figure 6, left) and a fully connected mesh (Figure 6, right). The two building blocks can be combined to form meshes of meshes, trees of meshes, meshes of trees, etc. (Figure 7). The process can be repeated as often as required to build a suitable topology. The topology, however, cannot have any loops, as the routers are not currently capable of detecting this condition.

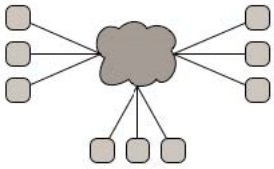


Figure 7: Combinations of the basic building blocks used to generate advanced routing topologies.

Mesh Router Architecture

The mesh routers developed for RTI-s adopted many of the design decisions made in the SF Express project. The router triad concept is perhaps the most obvious of the design decisions from SF Express, providing an elegant method of avoiding "message looping" in the mesh, while allowing an arbitrary number of routing decisions to be made when transferring messages. However, significant design changes have produced a radically more advanced and flexible infrastructure.

Flow Control

A tight flow control with Request to Send / Clear to Send (RTS/CTS) behavior was used in the SF Express design. SF Express used the mesh routers only within a single SPP, where latency was extremely low and available bandwidth greatly exceeded expected message transfer rates. The overhead of sending the RTS and CTS messages would not negatively impact the performance or scalability of the system. The communication medium of choice (MPI) requires pre-posted receive buffers of a known size, requiring a RTS/CTS protocol for sending large messages. However, recent trends have shown CPU power improvements far outpacing network latency and bandwidth improvements. On modern networks, a RTS/CTS protocol poses a significant performance burden. Therefore, the Mesh Router architecture has an eager send protocol with messages dropped by priority when queues overflow.

Application-Independent "Message" and "Interest" Objects

The Mesh Router software is object-oriented (C++), with a limited number of standard interfaces to "user message" and "interest" base classes. For present purposes, the implications of this factorization are:

- The Mesh Router system is designed to be compatible with ongoing changes and evolution within the RTI-s system, requiring little more that "re-compile and re-link".
- The Mesh Router system can support applications other than SAF/RTI, given appropriate different instances of the message and interest objects.

Simplified, General-Purpose Router Objects

The many distinct router varieties ("Primary", "PopUp", "PullDown", "Gateway") of the SF Express router network have been replaced by a single router object, as indicated by the schematic in Figure 8. Routers simply manage interest-limited message exchange among a collection of associated clients. The distinctions that had been hardwired into the various router types of SF Express are now summarized by sets of flags associated with the clients. The flags (simple boolean variables) specify whether:

- Client is a source of data messages.
- Client is a sink of data messages.
- Client is persistent (non-persistent clients are destroyed if the communications link fails).
- Client is "upper" or "lower" (this simple hierarchy provides the mechanism to prevent message cycles).

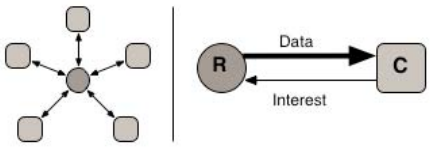


Figure 8: High level schematic of a router process (left) and dataflow of router/client connection (right).

These four flags are sufficient to reproduce the

specific communications model of Figure 4 and a number of other networks, such as the tree router model available in the JSAF/RTI-s library, and the schematic Tree/Mesh mixture of Figure 7.

Factorized Communications Primitives

The Mesh Router object design relies on a very careful isolation/factorization of the underlying message exchange protocol from the rest of the software. The essential object design is indicated in Figure 9 and has three layers:

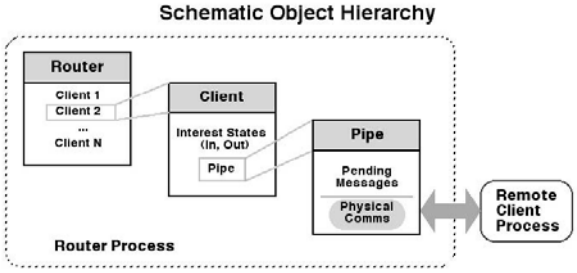


Figure 9: Schematic design of the Mesh Router application.

Router Objects: These are little more that smart lists of objects associated with the clients in Figure 9. In normal operations, routers simply execute the fundamental message and interest manipulation methods for the associated clients. Routers are also responsible for management of the overall client list, including:

- Removal of clients that have stopped communicating.
- Initiation of communications links, as needed, to specified (persistent) clients.
- Client additions, in response to requests from external processes.

Client Objects: Managers of the interest declarations and pending message queues for each (external) client process.

Pipe Objects: The interface between the Message / MessageList formalism of the Mesh Router software and the real world "bits on the wire" communications to the actual external processes. The Pipe object base class provides the last essential factorization of application specific details from the overall, general Mesh Router framework.

The communication factorization within the Pipe class is essential to the general applicability and ease of use of the Mesh Router system. A number of specific Pipe classes have been implemented to date, with the most important being:

- RtisPipe: Message exchange using the RTI-s framework. (Indeed, this object has been built entirely from objects and methods in the RTI-s library).
- MemoryPipe: Message "exchange" within a single process on a single CPU. This is used when two or more router processes in the sense of Figure 8 and Figure 9 are instanced as distinct objects within a single management process on a single CPU.

The factorization of application-specific communications mechanisms is, in fact, slightly more complicated than just indicated. The Pipe object has sufficient virtual interfaces for data exchange between a router and a general client. An additional virtual object/interface (the “ConnectionManager”) is needed to support dynamic addition and deletion of clients during router operations.

Router Configurations/Specifics, This Work

The numerical experiments described in this work explore two different overall communications topologies built from basic Mesh Router objects: the "Tree" and "Mesh" topologies shown in Figure 10.

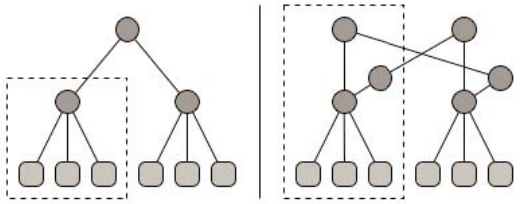


Figure 10:Basic Topologies Available using the Mesh Routers.

In the Tree topology, there is an entire CPU allocated to each router. All connections (simulator to Router or Router to Router) use the full RtisPipe instance. The persistent router clients in the sense of Section II are the upper router clients (if any) for each component router. All other communications links are generated dynamically.

For the Mesh Topology simulations, all three routers within the basic triad of Figure 4 are instanced as distinct Router objects on a single

CPU, with MemoryPipe connections are used for the Primary PopUp and Primary PullDown links within a single triad. All other links in Figure 10 use the RtisPipe, with the cross-triad PullDown Primary links persistent.

As noted, the current RtisPipe implementation is based entirely on objects and method calls within the current RTI-s library. This is important for demonstrating "ease of insertion" of the Mesh Router formalism into the RTI-s libraries, but it does result in a few minor inefficiencies. These include one extra memory copy per message and duplicate "interpretations" of incoming interest declaration messages. These inefficiencies can be removed in future, more finely tuned Pipe instances. Indeed, the careful communications factorization within the Mesh Router package supports mixed Pipe instances tailored to communications specifics for any of the individual links in Figure 10. In particular, the optimal Pipe instances for WAN and LAN links may be quite different. Though supported by the overall design, these refinements are beyond the scope of this particular paper.

Results

The Koa cluster at the Maui High Performance Computing Center was utilized for testing the Mesh Routers. Koa is a 128 node Linux cluster with two 3.06 GHz Intel Xeon processors and 4 gigabytes of memory per node. Nodes are interconnected via gigabit Ethernet. All routing topologies were generated using the standards for the JUO experiment: 5 federates per router and 4 routers per router (the second only applicable to tree routers). The default configuration parameters were used for both RTI-s and the Mesh Router. Since the Mesh Router utilizes the RTI-s communication infrastructure, we believe that any parameter tuning done to one system would apply equally well to the other system. To highlight the importance of topology in routing infrastructure, we show the Mesh Routers running in a tree configuration in addition to the standard RTI-s tree.

A number of tests ensured the Mesh Routers performed as required for JSAF experiments. The mesh infrastructure was used for an extended simulation using the JSAF suite. As expected for a small-scale simulation, the Mesh Router and RTI-s tree router were indistinguishable to the JSAF operator.

Latency measurements were taken on the Koa cluster. The Mesh Router performed slightly better in mesh configuration than in either tree configuration, but were within the measured error. Koa's low latency network combined with a short tree (only 3 levels deep) account for this measurement.

System Throughput

For testing the maximum throughput of the routing infrastructures, pair-wise communication was used. Attribute updates were sent between process pairs as fast as possible, with loose synchronization to ensure multiple pairs were always communicating. The average per-pair throughput, specified in number of reflectAttributeValues()calls per second for a given message size, is shown in Figure 11. For the test, 50 pairs were utilized, with 28 tree routers or 20 mesh routers creating the router infrastructure.

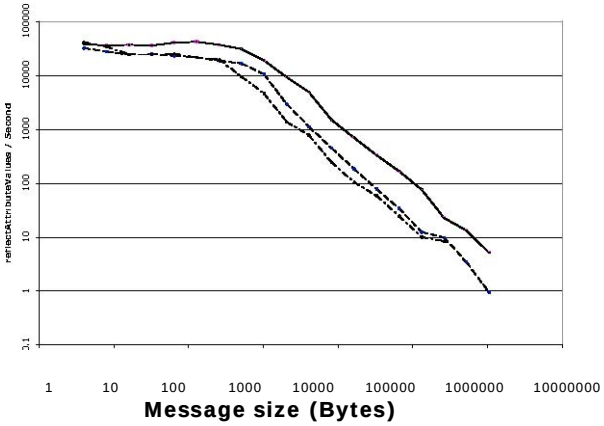


Figure 11: Realizable point-to-point bandwidth full communications load

As expected, Figure 11 shows that the maximum number of updates per second goes down as message size increases. The mesh router in a mesh configuration is able to move more traffic, and thereby cause more updates than either the RTI-s tree infrastructure or the Mesh Routers mapped into a tree topology. The RTI-s and Mesh Router tree configurations both would slow down at the root node of the tree, causing both lower realized aggregate bandwidth and an increase in dropped messages as message queues increased in length.

The RTI-s tree router performed much better than the Mesh Router in a tree configuration. This is not unexpected, as RTI-s has been finely

tuned to reduce memory copying and contention. The Mesh Router lower level has only started to be tuned for optimal performance on a Linux system. We see no implementation detail that would prevent the Mesh Router from matching the performance of the RTI-s routers and believe that further tuning will increase the performance of the Mesh Router in any configuration. federate to contact every other router in its mesh. The solution is to use a hypercube or similar topology, which provides scalable broadcast capabilities while maintaining bisectional bandwidth. The work required to develop such a topology should be minimal, with most of the effort spent on reducing the work required to specify the topology.

Future Work

The mesh routers currently provide a scalable solution for message routing in an RTI-s based federation. Future work will focus on fault tolerance, performance tuning, and investigation of supporting a fully compliant RTI implementation.

We have taken care to design a system that should allow plug-in adaptation to any RTI with a point-to-point communication infrastructure. Provided the client bounding assumptions are followed, the scalability shown for RTI-s should also apply to other RTI implementations. It is important to note, however, that a federation relying on timestamp message ordering will not see increased scalability with the Mesh Router architecture. . Timestamp ordering requires all-to-all communication, placing enormous stress on the communication fabric. Previous experiments have shown abysmal scalability (Fujimoto, 1998) and the authors see no reason to expect any improvement using a mesh topology.

As the size of a simulation increases, the chance of failure in the network or hardware increases. With the ever-increasing size of simulations, the ability of the routing infrastructure to handle failures is becoming critical. The routers handle very little state, so the data loss when a router fails is not critical. However, until the router is restored, messages will not be delivered properly. If the lost router is the connection point for a site, a large portion of the simulation is suddenly not available. One potential solution is to allow loops in the mesh topology. This provides N + 1 redundancy for the connections, as there can be multiple paths between sites. If one path fails, the system will adjust and use the other available paths. The long-term solution is to provide an adaptive, dynamically configuring topology that adjusts to failures and new resources. The basic Mesh Router objects could accommodate these generalizations.

There are some not-uncommon communication patterns for which the fully connected mesh is not well suited. One such pattern is a broadcast, which requires the router triad for the sending

Conclusion

The mesh router infrastructure presents a scalable routing infrastructure for both local and wide area communication. The routers are capable of being organized into a number of topologies, and should be easily extensible into new routing topologies. For wide area networks, the flexible routing topologies allow communication over all available network links, without the hub and spoke problem of the treerouters. Within a local area network, the mesh routers provide a scalable communication architecture capable of supporting hundreds of federates.

Acknowledgements

We would like to thank the Open Systems Laboratory at Indiana University, Aeronautical Systems Center Major Shared Resource Center, and the Maui High Performance Computing Center for the use of computer resources for performance measurement. We would also like to thank Bill Helfenstein for advice on integrating the mesh router code with the RTI-s code base. This material is based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Brunett, S., Davis, D., Gottschalk, T., Messina, P., & Kesselman, C. Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments. In Proceedings of the Heterogeneous Computing Workshop, pages 29 – 42. IEEE Computer Society Press, 1998.

Brunett, S. & Gottschalk, T., A Large-scale Metacomputing Framework for the ModSAF Real-time Simulation, Parallel Computing 24, 1998.

Calder, R. B., Smith, J. E., Courtemanche, A. J., Mar, J. M. F., Ceranowicz, A. Z. ModSAF behavior simulation and control. In Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation. Orlando, Florida: Institute for Simulation and Training, University of Central Florida, March, 1993.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J., (2002), Reflections on Building the Joint Experimental Federation, Proceedings of the 2002 I/ITSEC Conference, Orlando, Florida.

Dahmann, J., Olszewski, J., Briggs, R., & Weatherly, R. High Level Architecture (HLA) Performance Framework. Fall 1997 Simulation Interoperability Workshop, Orlando, FL, 1997.

Fujimoto, R. & Hoare, P., HLA RTI Performance in High Speed LAN Environments. Fall Simulation Interoperability Workshop, September, 1998.

Lucas, R. & Davis, D., Joint Experimentation on Scalable Parallel Processors. In Interservice/Industry Training, Simulation, and Education Conference, 2003.

MPI Forum. MPI: A Message Passing Interface. In Proceedings of 1993 Supercomputing Conference, Portland, Washington, November 1993.

Defense Modeling and Simulation Office. High Level Architecture Interface Specification, v1.3, 1998.

Rak, S., Salisbury, M., & MacDonald, R., HLA/RTI Data Distribution Management in the Synthetic Theater of War, Proceedings of the Fall 1997 DIS Workshop on Simulation Standards, 1997.

Appendix H

Successful Joint Experimentation Starts at the Data Collection Trail—Part II

Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao
M&S Team, Experimentation Engineering Department, J9 USJFCOM
Suffolk, Virginia
rgraeben@ida.org, grafuse@alionscience.com, rmiller@alionscience.com & kyao@isi.edu

ABSTRACT

Last year Joint Forces Command's, Joint Experimentation Directorate (J9) initiated planning and development in technical support of the most complex experiment (URBAN RESOLVE) undertaken to date. The experiment trials (Summer 2004) will explore future concepts and technologies for achieving situational awareness and understanding when operating in a robust large-city urban environment. In addition, the need for generating quantifiable results took on a renewed level of interest. The Commander, Joint Forces Command directed that future experiments provide findings that can survive critical scrutiny, particularly if those transformational products and solutions are to be promulgated across the Department. The authors' add another chapter to last year's paper, as they craft a system for providing more creditable and quantifiable data to support experiment findings. This paper will cover: changes made in the initial plan for data collection and analysis as new challenges arose along the way; the technical issues related to the architectural choices; as well as the challenges awaiting the group of individuals charged with maintaining a nationwide, distributed federation and network whose ultimate goal is to provide cogent, traceable data generated from the federation and human-in-the-loop player inputs. In preparing for the experiment trials, initial data storage assumptions gave way to the realities of finding more robust methods of collection as bandwidth traffic increased as federation architectures were modified to support emerging user requirements. Innovative approaches on how near-real-time data would be collected were instantiated as attention turned towards the post-processing needs that would sustain the experiment analysis team in the months following the trials. Integrating scalable parallel processors and addressing issues dealing with the means for storing and retrieving extremely large quantities of data added to the challenges. Finally, major lessons learned will be addressed from a transformational perspective.

ABOUT THE AUTHORS

Bob Graebener, Colonel, United States Army (Retired), has been a member of the professional staff at the Institute for Defense Analyses (IDA) since March 1997. He is currently a Research Staff Member at IDA and has primarily involved in supporting JFCOM J7/J9 in Modeling and Simulation related matters for the past nine years. He is currently working towards a doctoral degree in Systems Engineering from GWU.

Gregory Rafuse is a data collection analyst and developer and is currently the lead developer for the data collection toolkit. He is a Software Engineer with Alion Science and Technology. Mr. Rafuse has previously served seven years with the US Army as a Field Artillery Crewman. He also possesses an AAS in Computer Information Systems (CIS) from McLennan Community College and is pursuing a BS in CIS from Strayer University.

Robert Miller is a Senior Software Engineer with Alion Science and Technology. He brings over 11 years of experience to the current effort of designing, coding, and testing software for the Future After Action Review System. He holds a Bachelors Degree in Engineering from The Cooper Union School of Engineering and a Masters Degree in Computer Science from the City University of New York.

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University. For his Ph.D. thesis he implemented a spatial and physical reasoning system that automatically generated grids for novel geometries for computational fluid dynamics simulators.

PRELUDE

The reader should be aware that the title of this paper ends with "—Part II". Those familiar with human-in-the-loop simulations like JSAF (Joint Semi-Automated Forces) and joint experiments set in the year 2018, such as URBAN RESOLVE, realize that when one pushes the boundaries of simulation-support-to-experimentation a discovery process, in its own right, is created as the bounds of "what can be done in simulation" is continually challenged and superseded. Over the course of the past year, what started as a concept for developing "a best approach for collecting and analyzing data" gave way to the practical experience gained through the number of integration events necessary to prepare for the formal trials. The authors' felt it necessary to add another chapter to last year's journey (Graebener, et. al., 2003).[1]

INTRODUCTION

The initial concept of how to approach data collection and analysis when faced with a simulation federation that could generate data records in the terabyte range has evolved over the past year. Whereas PART I laid out the challenges associated with extremely large data generation conditions and the initial approach for meeting the experiment data collection requirements, and significant detail of the major changes will follow.[2]

This paper will cover:

- Changes made in the initial plan for data collection and analysis as new challenges arose along the way, as well as technical issues related to the architectural choices;
- Subsequent modifications in the data analysis tools to meet the changing user requirements,
- Challenges awaiting the group of individuals charged with maintaining a nationwide distributed federation and network whose ultimate goal is to provide cogent, traceable data generated from the federation and human-in-the-loop player inputs.
- Finally, lessons learned will be addressed from a transformational perspective.

In preparing for this year's experiment trials, initial data storage assumptions gave way to the realities of finding more robust methods of collection when network traffic increased as federation architectures were modified to support changing/emerging user requirements. Innovative approaches on how near-real-time data would be collected were instantiated as attention turned towards the post-processing needs that would sustain the experiment analysis team in the months following the trials. Integrating scalable parallel processors and addressing issues dealing with the means for storing and retrieving extremely large quantities of data added to the challenges. (Table 1)

Table 1. Data Collected During Dress Rehearsal Week.

# of Data Records of Interest (stored in 576 tables)	Percent of Total Data Logged	Size of Database
264 million	15-20	45 GB
Average size of each record: 180 bytes		

BACKGROUND

The original concept behind the FAARS (Future After Action Review System) toolkit was based on utilizing commercial-off-the-shelf (COTS) products for collecting simulation data. The original design specifications for the FAARS toolkit comprises three separate modules; a Data Collection Module utilizing hlaResults as the federation data interceptor and storage transport, a Near Real Time Module utilizing MySQL as the data storage medium along with a Apache web server with PHP scripts as the data presentation and analysis medium, and a Post Event Analysis Module using MySQL as the data storage medium and a custom written C++ user interface for accessing stored data for processing and analyses. Although this design works well and is being used in several joint experiments, it was not robust enough to support the URBAN RESOLVE series. Initial testing

[1] Last year's paper will be referred to as PART I for the remainder of this paper.

[2] The authors recommend a review of last year's paper to serve as a point of departure. Go to: http://www.alionscience.com/pdf/Data_Collection.pdf

results using the complex urban terrain and tens of thousands of entities being detected by a large constellation of sensors were adequately handled using Scalable Parallel Processor clusters, however the methodology of using hlaResults as the data collector no longer met the requirements. The reasons for replacing hlaResults were:

1) hlaResults only works with an NG-style RTI. For the UR effort, we are using an s-style RTI, which is a different implementation loosely based on DMSOs RTI-NG v1.3 standard.

2) When hlaResults subscribes to ALL entity traffic this overwhelms the physical network interface and causes packets to be dropped at the physical interface, effectively "missing" information.

3) Due to the nature of how cluster computers function, a significant amount of the simulation event information could not be effectively be logged.[3] Based on these factors, a different data logging architecture was needed.

INTERCEPTOR/LOGGER

The Interceptor/Logger application, an early version described in PART I, is an application process that resides on individual simulation nodes within the federation.[4] The determining factor on where to utilize the mechanism is determined by which federates are publishing information needed for data collection. The interceptor/logger, utilizing functionality in the RTI Application Programming Interface, inserts "hooks" into the published data streams by the RTI and then splits off two child processes; one process that writes and compresses the intercepted data into binary "log" files and a second process that decodes the data stream and inserts the decoded data into an embedded database application called SQLite. A separate daemon process called "sqlited" handles incoming socket-based connection attempts to query information that has been stored in the local database. Figure 1 is a diagram of the process.

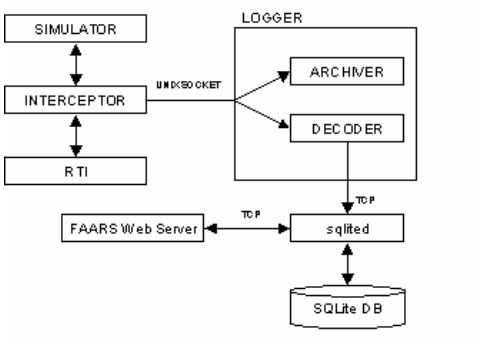


Figure 1. Interceptor/Logger Process

Because of the methodology of running interceptor/loggers on each simulator with data of interest, a separate mechanism was needed to retrieve information stored at each simulator location. A separate application process called "Aggregator" was developed that would handle the intercommunication between simulators logging data. The Aggregator is configured in a tree-like fashion, with a "Root Aggregator" at the head of the tree and "Child Aggregators" in branches from the root. The various branches reach out to the individual leaf instances of "sqlited" on each simulator. The interface to the Root Aggregator takes a Structured Query Language-formatted query and passes the query on to each branch Child Aggregators until the query finally reaches the individual instances of "sqlited". As each instance of "sqlited" responds with the requested data for the query, the Child Aggregators assemble the returned information in order of response and forwards the data on to the Root Aggregator which then assembles the complete returned information and forward it on to the original requestor. The Aggregator model works on Transmission Control Protocol (TCP) socket-based connections between the Root Aggregator and subsequent Children Aggregators.

Near Real Time Retrieval Of Data

With the utilization of the ISI interceptor/logger, the possibility of retrieving simulation information in a "near real time" manner became a reality. Typically, data collection efforts have had to wait until after collected logger files have been processed before any specific event information could be derived. This is a vast improvement in functionality and provides a wide range of uses that are still being realized as we move forward in the software development effort.

The Near Real Time data retrieval effort is based around the ability to query the ISI interceptor/logger application, retrieve the logged information from each node and store the retrieved information into a local

[3] The data could not be intercepted and logged by the hlaResults product because a significant amount of simulation traffic would be exchanged between SPP cluster nodes running the simulation and not transmitted outside of the cluster, a necessary prerequisite for hlaResults.

[4] Developed by the Information Sciences Institute (ISI) at the University of Southern California,

Relational Database Management System (RDBMS). The retrieved information is then used by the FAARS Near Real Time web server interface to allow users of the system to view various reports, charts and graphs based on the available information.

The process of retrieving intercepted information from each of the active ISI interceptor/loggers is handled by a series of BASH shell scripts on the FAARS web server. Each BASH shell script is targeted towards retrieving specific information, such as entity object states, and is used to process the retrieved information into the local RDBMS (aka cache). The data retrieval process is based on three steps. The first step is to send the request for information to the Root Aggregator. The methodology used by the retrieval process is based on making a TCP socket-based connection to the Root Aggregator and sending an SQL-formatted query. The second step is to wait for a response and process/validate the retrieved data and write this data to a temporary file. The expected response back from the Root Aggregator is a stream of plain ASCII text, which is tab-delimited for fields and is carriage return delimited for individual records. This information is then written to a temporary file in this same tab-delimited/carriage return delimited format. The third and final step is to then load the temporary file's data into the local cache.

The FAARS web server RDBMS cache uses MySQL v4.1.1. as the database engine. The database schema for the cache is based primarily on the schema used by the ISI interceptor/logger. This helps in facilitating compatibility with the information that is being utilized in near real time and data being reviewed post event. The main difference between near real time and post event processing is the different indexing schemas utilized on the local cache. The indices applied to the local cache database have been specifically tuned to support the types of queries that the FAARS web server uses for data displays.

Storage Space Requirements

When the overall design of the FAARS toolkit began to change to utilize the ISI interceptor/logger, physical storage space for collected data files and consolidated database became an issue. With the switch to using a larger-scale database engine than previously used and the need to analyze larger amounts of data than previously anticipated, the need for more physical media space became apparent. Where it was once thought that ten's of Gigabytes (GB) of storage space would be sufficient, it soon became apparent that this was not going to be acceptable. The central importance of disk space is its centrality to all three aspects of the process: storage of compressed logger files, storage space for staging uncompressed logger files while loading into consolidated database, and space needed for the final database tables and indices. What was finally settled on was a RAID 5 disk array totaling 1.7 Terabytes (TB) of disk space with a stand by disk array of 1.3 TB in size.

Because of the distributed nature of logging data that has begun to be utilized, it has become necessary to develop means to: retrieve all of the saved binary data logs on each simulator where the ISI interceptor/logger was instantiated; prepare and decode the binary data files, and then; insert the decoded data into a consolidated database representing the complete accumulation of data for a particular event. A process called "Data Staging" has been developed that accomplishes these tasks in an organized, efficient manner, making the best usage of available bandwidth, processing cycles and disk space. (See Figure 2) The Data Staging process begins with retrieving the binary log files at the end of each day's simulation run from each simulator logging data. The data is moved and stored on the local storage point in a hierarchical format based on the event name, day of the event and the simulator where the log file was retrieved. Once the data has been moved, Perl-based scripts are run against the individual binary log file to decode and format the binary data into plain-text, comma-separated value (CSV) flat files. The translation of the data and the creation of the storage database schema are based on utilizing definitions found in the Federation Object Model (FOM) and Federation Execution Document (FED) for the federation in use. Each CSV-formatted file represents a section of data to be inserted into the consolidated database for the event. A final Perl-based script takes the CSV-format files and inserts the decoded data into the appropriate table within the consolidated database.

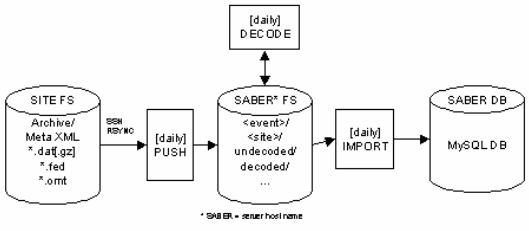


Figure 2. Data Staging Process

Database Engine Configuration Issues

Previously, the MySQL v4.0.18 RDBMS was selected for storing the decoded logger data for Post Event analysis operations. As a database engine, MySQL is

both an open-source and a commercial product line with a significant amount of engine performance tuning available for the end user to adjust based on specific needs. Through trial and observation, several adjustments to the database engine were decided on that would afford us the best performance for both the loading and the retrieval of data.

Database Table Configuration Issues

The MySQL database server engine supports several different table types, the default being MyISAM but provides support for BerkeleyDB, InnoDB, MERGE and MEMORY table types. It was decided to stick with the default type of MyISAM mainly for the fact that once data is loaded into a table, the data in the table becomes static and read only. Both BerkeleyDB and InnoDB table types are transaction safe, which for our purposes are not necessary.

One of the primary concerns for the table definitions revolved around the number of rows the tables will contain. During initial testing, it was discovered that the default number of rows that a MyISAM table could hold was less than the number of rows to be loaded. The overall data size for the table is determined by the types of fields used for the table and the data size for each type. Examples of this would be an INTEGER field type, which can have a data size up to 4 bytes and VARCHAR field type, which can have a data size up to the length of the text value + 1 byte. For the purposes of this experiment, there are three table settings that had to be set in order for the tables to scale to the number of rows anticipated. By adjusting the AVG_ROW_LENGTH, MAX_ROWS and ROW_FORMAT variables for MyISAM tables, it was possible to adjust the number of rows of data that the table can have. The ROW_FORMAT variable defines how the table rows should be stored. The option value can be FIXED or DYNAMIC for static or variable length row formats. When a table is defined that does not have BLOB or TEXT type columns, you can force the table format to FIXED or DYNAMIC with the ROW_FORMAT table option. This causes CHAR and VARCHAR columns to become CHAR for FIXED format or VARCHAR for DYNAMIC format. The AVG_ROW_LENGTH variable defines an approximation of the average row length for a table. This should be set only for large tables with variable size records. With a MyISAM table type, MySQL uses the product of MAX_ROWS times AVG_ROW_ LENGTH to decide how big the resulting table will be. If neither of these variables is specified, the default maximum size for a table will be 4GB. Overall, adjusting these variables are an absolute must to support the number of rows of data that have been observed for both the Near Real Time cache database and the Post Event consolidated database.

Because of the number of rows of data being stored into tables, it is imperative that efficient indexing be applied based on a thorough analysis of how data is extracted from the databases.

All columns used are not necessarily indexed, but only the columns that would enhance a typical query. As an example, within most tables there are VARCHAR fields that hold RTI-determined object name values. The object name in question, for the most part, uniquely identifies a specific entity within the simulation. Any column in a table that contains this type of data has an index applied to it because most of the queries posed utilize this column type as part of the qualifier of an SQL statement. Other indices are applied on a table-by-table basis within both the Near Real Time cache and Post Event databases geared towards their unique needs, but painstaking research went into selecting the most efficient usage of indices as part of each database schema creation effort.

NEAR REAL TIME PROCESSING

An example of one of the tools used for near-real-time analysis is the Track Matrix. The track matrix table provides a tabular snapshot of the current (based on the last 30-35 minutes of simulation time before the query is submitted) number of tracks associated with each type of entity. The row labels of this table are the actual truth types of entities being tracked at the current time. The column labels represent the perception of the entities being tracked. The column headings are exactly the same as the row headings because the set of possible perceptions is the same as the set of possible track types; Perceptions are determined by the SLAMEM simulation federate's sensor fusion center utilizing algorithms based on Bayes Rule. The resulting target type with the highest probability is the type associated with the track. The table entry in a given row and column is the number of tracks belonging to the corresponding row type that are perceived to be the given column type. A column labeled 'Ambiguous' indicates that those tracks are not resolvable. This means that the sensor fusion process determined that two or more target types were equally probable as the type of target being tracked.

Subsequent details associated with the Track Matrix are a series of additional tables and graphs segregating tracks into their ages, which is defined as the length of time between when the track was created and the last time it was updated. By segregating tracks by their

age, it is possible to get a sense on how well sensors and, in some cases players, are aware of the entities being played within the simulation. Older tracks can be perceived as having a higher probability of positive identification as opposed to tracks that persist for a shorter amount of time.

POST EVENT PROCESSING

In accordance with numerous authorities, the highest-level decomposition of the Post Event Processing system was into a single control class, entity classes, and an interface class (inasmuch as the interface class was a straightforward application of Microsoft Foundation Class (MFC), it will not be discussed). There are three general entity classes, called Database, Processor, and Final_Results. These roughly correspond to a traditional functional breakdown into input, transformation, and output. To promote the greatest possible generality, interactions between the Database class and the other classes were performed using Open DataBase Connectivity (ODBC). The Final_Results class encapsulates Microsoft Excel or a commercial graphical package called ChartDirector. Communications to and from that class uses either Microsoft's OLE Automation or ChartDirector's API. The Processor class, as well as most of the infrastructure of the system, was written using C++.

The post processing system comprises eight overall functional areas, all invoked by the user. These functions are as follows:

1. A Killer/Victim (K/V) Scoreboard,
2. A Killer/Victim details display,
3. An Entity Life Cycle summary screen,
4. An Entity Details Display,
5. A Sensor/Target (S/T) Scoreboard,
6. A Sensor/Target details display,
7. A Track Perception Matrix, and
8. A timeline (String) depiction that displays, graphically, the events in the lifecycle of any specific entity.

K/V Scoreboard
As currently coded, the K/V Scoreboard is produced by querying for the number and enumerations of all killers and victims are obtained via simple SQL queries. For each possible combination of killer and victim, the Damage Assessment interaction is queried to obtain the relevant victim's state. This is recorded, along with the entity causing the damage. Summations are performed by type (as indicated by enumeration values). The final results are presented in the form of an Excel spreadsheet or comma-separated value flat file.

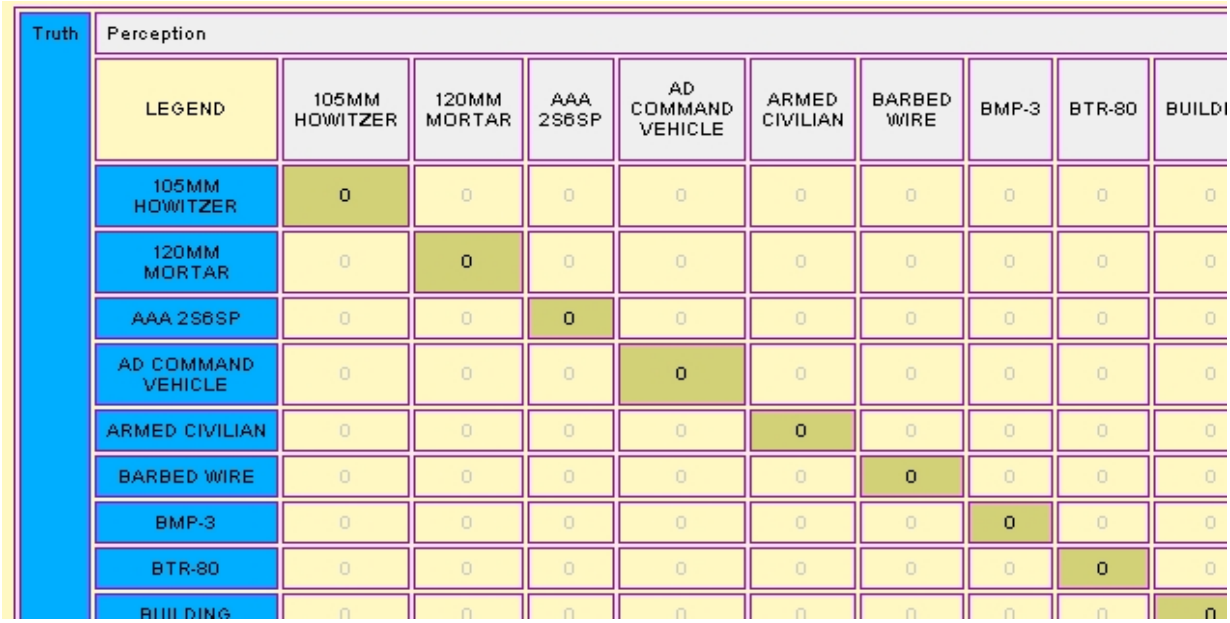
K/V Details Display
To obtain the details of any Killer/Victim interaction, the user is first presented with a screen enabling him to choose a particular killer and victim. Queries are performed against a lookup table to transform these English names into enumerations. An SQL statement is then constructed and executed that extracts the relevant fields from the Damage Assessment interaction.

Entity Life Cycle
The Entity life cycle summary output is derived largely from the entity state objects. All the entities used in the execution are gathered together into a vector. For each entity thus obtained, its entity state object is queried to obtain the fields necessary to compute its final state. This state is then determined and added to a running total.

Entity Details Display
Entity life cycle details are obtained from numerous objects. The user first chooses an entity via a series of drop-downs. To obtain the entity's state changes, its entity state objects are scanned for all records indicating a change of state, whose details are then recorded. The appropriate objects are then queried to ascertain the entity's creation and deletion details; data on sensor hits and weapon fire events, and detonations occurring on or near the entity.

S/T Scoreboard
The Sensor/Target Scoreboard summary output is similar in structure and layout to the Killer/Victim Scoreboard with the exception of the data obtained for the matrix display. For each possible combination of sensor platform and detected target, the Contact Report interaction is queried to obtain the relevant information concerning the detected target and the functional mode the sensor used to interrogate the target. Summations are performed by sensor platform and by sensor mode (as indicated by enumeration values) with the final results being presented in the form of an Excel spreadsheet or comma-separated value flat file.

S/T Details Display
To obtain the details of any Sensor/Target interaction, the user is first presented with a screen enabling him to choose a particular sensor platform and detected target. Queries are performed against a lookup table to transform these English names into enumerations. An SQL statement is then constructed and executed that extracts the relevant fields from the Contact Report interaction.



Truth	Perception								
	LEGEND	105MM HOWITZER	120MM MORTAR	AAA 2S6SP	AD COMMAND VEHICLE	ARMED CIVILIAN	BARBED WIRE	BMP-3	BTR-80	BUILDI
	105MM HOWITZER	0	0	0	0	0	0	0	0	0
	120MM MORTAR	0	0	0	0	0	0	0	0	0
	AAA 2S6SP	0	0	0	0	0	0	0	0	0
	AD COMMAND VEHICLE	0	0	0	0	0	0	0	0	0
	ARMED CIVILIAN	0	0	0	0	0	0	0	0	0
	BARBED WIRE	0	0	0	0	0	0	0	0	0
	BMP-3	0	0	0	0	0	0	0	0	0
	BTR-80	0	0	0	0	0	0	0	0	0
	BUILDING	0	0	0	0	0	0	0	0	0

Figure 3. Track Perception Matrix

Track Perception Matrix

The Track Perception Matrix summary output is designed to show information concerning simulation Tracks and how they are being perceived by the sensor model being used by the federation. For each possible combination of true entity types (truth guise) and perceived entity types (perceived guise), the Track and Track_probabilities interactions are queried to obtain the relevant information used to generate the display. The display consists of column headings representing the perceived guise possibilities, row headings representing the truth guise possibilities, and a diagonal across the table which represents where the truth guise and perceived guise intersect. See Figure 3 for an example section from a Track Matrix. The row and column heading extents are determined in advance by aggregating entity types together via a lookup table. The information displayed is then available for export to either an Excel spreadsheet or comma-separated value flat file.

String Chart

The String chart requires much of the same data as contained in the Entity Life Cycle details screen, and therefore uses the similar algorithms to gather data. However, instead of sending the results to an Excel spreadsheet, the data is fed to a commercial graphing product (ChartDirector). This product produces a timeline whereby events are depicted as color-coded icons. Placement of the icons at different y-axis values depicts the different events. These are placed in proper time order, with the x-axis showing wall clock time.

The String chart allows the user the choice of displaying the requested information in three possible ways with the data segregated into four sections:

- Entity Event,
- Blue Activities,
- Track Events, and
- Sensor/Target Events.

The Track Events and Sensor/Target Events sections are also segregated to show all instances of individual track numbers and sensor name/mode combinations or actual target type and bumper number. Each section or subsection is then sorted by time.

The Entity event section graphically depicts all changes that are derivable from an examination of objects received via the High Level Architecture (HLA) federation. These include entity creation, entry into damage states (firepower, mobility, firepower and mobility, or total destruction), moves and stops, and entry into or exit from camouflage. This section also records instances of the entity firing its weapon, receiving incoming fire, being deleted, or being recreated after a deletion.

The Sensor/Target section can depict two types of information. If the entity in question is being sensed, it will contain a comprehensive display of all the sensors that “saw” the entity, the sensor’s mode, the highest

acquisition, the highest correct perception, the perceived type, and the time of detection. If the entity was within the footprint of the sensor but was not detected, a brief explanation of why is given.

If the entity in question is itself a sensor, the chart displays a listing of all entities that were detected, their actual type and bumper number, the highest acquisition, the highest correct perception, the perceived type, the sensor mode employed, and the time of detection. As before, if the entity was within the footprint of the sensor but was not detected, a brief explanation of why is given.

The track events sections show all tracks associated with the entity. For each track, a listing of the top three most probable entity types is given, along with the computed probability of the entity being of that type and the number of sensor hits used to determine it.

The blue activities section is reserved for human actions, such as planning a mission, assigning it a priority, initiating an attack or mission abort, requesting a bomb damage assessment, etc. All such user actions are sorted by time.

In addition to the pictorial generated by ChartDirector, a file containing the corresponding raw data (in either CSV flat file form or as an Access database) is also generated to allow the analyst to examine the information used to generate the picture directly.

CHALLENGES

General Gordon Sullivan (Chief of Staff of the United States Army, 1991-95) once said, "You don't know what you don't know," a statement that accurately describes today's challenges in mining extremely large databases.

Some of the challenges under initial assessment by the FAARS team:

- Policies and procedures for allowing interested government agencies access to the data generated by the URBAN RESOLVE experiment.
    - The data storage server is connected to the DREN. *What is the best approach for allowing others on-line access while minimizing the impact on the UR data collection and analysis effort?*
    - *Will the answer be purely policy driven or can software and hardware solutions enable the simultaneous utilization of the database?*
- Less than twenty percent of the collected data is of primary importance to the data analysts, at present.
    - *What impact will occur when the remaining eighty percent of the data is transformed through the data staging process and available on the terabyte storage device?*
    - *Will new techniques be required?*

CONCLUSION

The discovery process is not solely a characteristic of the joint experiment, but touches many aspects associated with the experimentation effort. As Joint Forces Command and the Joint Advanced Warfighting Program at IDA demand more from the simulation community the ripple affect moves throughout the various federates providing support.

In this specific case, the data collection and analysis effort has met the near term challenges brought about by an experiment scenario that requires over one-hundred-thousand entities; 1.8 million buildings and man-made urban structures to set the stage for achieving situational awareness in the 2018 timeframe. Use of hundreds of scalable parallel processors each logging the data generated during run-time was the impetus for the FAARS effort.

New challenges that have arisen since PART I was published will pale as we contemplate the challenges that we face for the upcoming year's trials. The good news story is the FAARS team, in fact the whole M&S team in J9, will continue to meet and overcome whatever challenges arise, and who knows, there might be another chapter awaiting.

ACKNOWLEDGEMENTS

This material is based in part on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Dehncke, Rae W., Graebener, Robert J. 2004. *Urban Resolve: Joint Experimentation Raises the Bar for M&S.* Orlando: IITSEC 2004 Paper.

Graebener, Robert J., Rafuse, Gregory, Miller, Robert, and Ke-Thia Yao. 2003. *The Road to Successful Joint Experimentation Starts at the Data Collection Trail*. Orlando: IITSEC 2003 Paper.

Appendix I

An interdisciplinary approach to the study of battlefield simulation systems

John J. Tran
Information Sciences Institute
Marina del Rey, CA 90292
jtran@isi.edu

Jacqueline M. Curiel
Behavorial Cognition
Marina del Rey, CA 90295
jcuriel@behavorialcognition.org

Ke-Thia Yao
Information Sciences Institute
Marina del Rey, CA 90292
kyao@isi.edu

ABSTRACT

There are numerous advantages for conducting computer simulations that model wartime operations, which belie the popularity in implementing them. Among them are: the ability to easily model the variables the researcher is interested in, the ability to control the experimental scenario, and the ability to add or change variables as the need arises. A simulation environment's success may be enhanced by considering questions not normally at the center of simulation research. These include, but are not limited to, the following: to what extent does performance in the simulation reflect performance during real life situations, to what extent does the learning that occurs during the simulation transfer to real time situations, and is the cognitive processes that operate during simulations similar to the ones that operate in real time situations? In considering the first question, experimental procedures may be used to identify whether simulation performance reflects real life performance. In considering the second question, one may note that research has shown that performance has been influenced by the learning context, which may or may not influence the transfer of learning that occurs from the simulation to the real time situation. In considering the third question, one must include attention, language processing, and memory, as well as problem solving strategies.

This paper will propose an interdisciplinary approach to the study of the wide spectrum of battlefield simulation systems such as JSAF, STOW, and JESPP. It will show that the approaches and implementation of these systems up until now have been grounded in the computer science discipline. We will explore what cognitive science has to say about the simulation driven approaches. Integrating the viewpoint of fields such as cognitive science can provide valuable insights as to the effectiveness of these approaches by substantiating the validity of the system and increase the fidelity of the synthetic to real life experience.

ABOUT THE AUTHORS

John J. Tran is a researcher at the Information Sciences Institute, University of Southern California. He received both his BS and MS Degrees in Computer Science and Engineering from the University of Notre Dame, where he focused on Object-oriented software engineering, large-scale software system design and implementation, and high performance parallel and scientific computing. He has worked at the Stanford Linear Accelerator Center, Safetopia, and Intel. His current research centers on Linux cluster engineering, effective control of parallel programs, and communications fabrics for large-scale computation.

Jacqueline M. Curiel is a researcher with Behavioral Cognition, where her work focuses on situation model theory. Previously, she taught at the University of Texas at San Antonio where she conducted research on working memory. She received her doctoral degree from the University of Notre Dame where her dissertation focused on the influence of spatial and entity information on the repeated name penalty. Her master's thesis, also from the University of Notre Dame, focused on spatial and temporal organization in mental maps.

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University.

INTRODUCTION

Problem description

JSAF is a modern system for conducting synthetic battlefield experiments. JSAF has adequately modeled organic and inorganic entity behavior, (the physical model) [Ceranowicz] but lacks adequate techniques for measuring players' internal behavior. However, taking into consideration the player's state of mind during the analysis stage would increase the fidelity of the system.

JSAF uses a network of computers to model forces and conflicts. JSAF includes entities, environmental behavior, such as weather, and terrain. In its current state, we have the tools and experience to model physical behavior which answers the question: what is going on with the system? One example of the tools is the ability to conduct extensive logging of all simulation states. We do not have the adequate tools to answer what is the state of mind of the players. How do they relate to their environment?

Meeting the above goals will help us understand and validate the effectiveness of: (1) Doctrine and policy, (2) Simulation tools and environment, and (3) Sensor technology. In the next section we consider how the methods of cognitive science may provide potential tools to meet our goals with the specific focus on situation awareness (SA).

Motivation

Cognitive science and applied cognitive science refer to the umbrella disciplines that seek to understand both how the mind works (cognitive science and applied cognitive science), and how such knowledge can provide useful applications (applied cognitive science). These disciplines include, but are not limited to, cognitive psychology, artificial intelligence, and cognitive ergonomics. By taking into consideration the capacities of what the mind can process, we hope to develop methods and models that more accurately assess and represent players' performance.

To do so, we will rely primarily on cognitive psychology and applied cognitive psychology because they are pioneers among these disciplines in subject matter and in the use of objective measures to infer behavior, they provide experimental frameworks that may be useful for testing theories, and they have existing cognitive models that may be used for comparison.

We are interested in the case of situation awareness because SA researchers address the same issues we are trying to address, namely what is the participant's state of mind and their relationship to the environment and how is this relationship quantified? In addition, the abstract definition of situation awareness does not favor any particular field, although in practice much research effort has been devoted to aviation, pilot, and emergency crew SA. We are proposing to extend this study to synthetic battlefield simulations in the context of cognitive psychology by providing a:

1. Background for situation awareness and JSAF simulation,
2. Taxonomy of situation awareness,
3. Draw upon cognitive methods to increase the accuracy of situation awareness measurement
4. Accurate measurement of situation awareness increases the ability to evaluate sensor effectiveness

BACKGROUND

Background on JSAF

JSAF, a joint semi-automated force simulation system that models battlefield environment, is federated [cite Williams2003] and has many components working together to create a synthetic battlefield and conflict simulation environment. These components, together, operate to model the JSAF system's physical and behavioral realism. The JSAF software serves as modeling and simulation tool for training and doctrine development purposes.

The most recent JSAF mission is the Joint Urban Operation (JUO) exercise, and amongst its many objectives are to (1) provide training and development urban warfare tactics, and doctrine, and (2) unify data

logging, (3) perform sensor platform validation, and (4) model and evaluate players' internal state of mind.

Needs driven impact	Development Efforts	Logging capability
Tactics and doctrine development & needs for training	Earlier development of sims	Early stage with minimal logging facility
Add realism Improvement to the system	Model Behavior & Sensors Technology	Disjointed logging facility
Higher fidelity	Scalable system (JESPP)	Unify Logging more than needed
Situation awareness	Analysis tools *Mental Models*	Analysis of log

Table 1: Evolution of behavior modeling and analysis in JSAF

Table 1 outlines a perspective on the progression of JSAF development driven by the needs to have data logging, complex tool development, and functional requirements. The Joint Experiment Scalable Parallel Processor (JESPP) and the Future After Action Result System's (FAARS) effort, leaded by a group at ISI and TEC, focuses on the ability to collect for the first time all simulation event data, and with this capability, the JSAF team approach the ability to measure situation awareness. Moving forward, understanding and measuring situation awareness in JSAF requires a cognitive perspective.

Background on Situation Awareness

Specifically area of intense research interest and although several definitions for the term exists one that is commonly accepted refers to SA as the perception and comprehension of surrounding environment that allows for a projection of the future states of affairs [Endsley]. The term *situation model*, which is distinct from mental model of how a system operates, has also been used to refer to SA.

In military terms, SA is a static spatial awareness of friendly and enemy troop positions [MHOB]. With regards to JSAF, it is not to clear to researcher whether or not JSAF has SA properties that correlate poor SA to poor planning, and therefore deprive players of comprehension and perception of simulation environment.

In JSAF, any relevant discussion of SA must be framed in the context of the concept of "cells" of which there are three: red (hostile), blue (friendly), and white (neutral omnipotent observer). Each cell is made up of a group of players on the same side (with the same military mission), and for red and blue cell, each of which has some level of collective situation awareness of the opposing force. In particular, the game-play objectives of the JUO exercise are: (1) for the red force to evade the blue force, and (2) for the blue force to capture the red force.

[include picture of blue & red cells pvd]

The geographical dimension of the area of interest (AOI) is enormous, and the blue players have access to sensor information that tracks the red force's activity. The white players have the views of both the red and the blue team, but serve as neutral observers and evaluators. In light of the above information machinery, the direct relationship between SA assessment and sensor effectiveness is yet to be determined.

THE CHALLENGE

Taxonomy of situation awareness

Domain Behavior Depth

There has been much work devoted to classifying SA and the various segments of military and aviation simulation. To this end, we focus on how an alternative taxonomy of SA affects the use of cognitive psychology methods for a better quantitative evaluation of players' internal state. Pew *et al* suggests the need to have in depth domain of behavior classifications, and breaks down as followed: (1) the organizational level SA, which is guided by doctrine and policy, and (2) the individual level SA, which is guided by tasks. He added that common to both, the process for obtaining information is based on psychology (Pew1997).

	Individual	Group
Micro model	F15 fighter pilot	C130 Crew
Macro model	JSAF red player	JSAF red cell

Table 2: Taxonomy of the Domains of SA

Figure XYZ: Example of Micro Model SA; analogous to first person shoot 'em gaming.

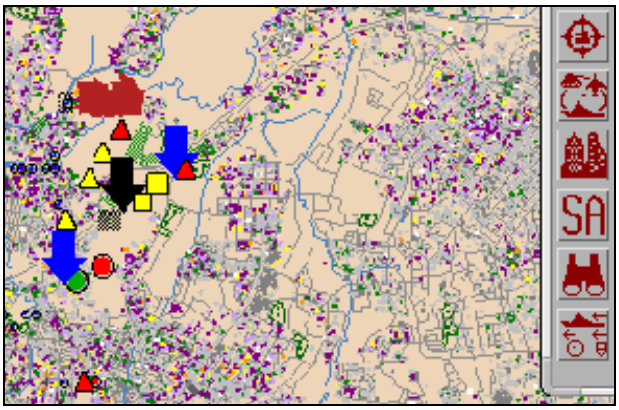


Figure XYZ: Example of Macro Model SA; this is a screen capture of how the red players place their Situational Objects on a PVD (game console).

We further expand on Pew's classification with a two dimensions: (1) along the first dimension, a distinction is made between "micro" and "macro" model of SA, and (2) along the second dimension, we retain Pew's distinction between individual and group SA levels. Together, they form the four quadrants of SA taxonomy. Table 2 illustrates some examples of the quadrants of SA taxonomy.

Table 3 highlights a comparison of characteristics between the micro and macro model. The intimacy between the players and their environment sets the distinction between the micro and macro model. Additionally, the locality of awareness also points out to another difference between the two models. The following examples speak more to the specifics of this classification.

Micro Model	Macro Model
Fine grain relationship between subject and environment	Coarse grain relationship between subject and environment
Higher local awareness	Lower local awareness
Lower global awareness	Higher global awareness
Greater psycho-physiological impact	Less psycho-physiological impact

Table 3: Comparison of Characteristics of the Domains of SA

Micro Model

The individual level SA of the micro model is defined as task oriented SA of participant with minimal communication flow, and the focus is on individual cognitive processes. For example, a pilot in a cockpit of an F15 fighter jet has an SA level that is only relevant to his or her environment, namely, the instruments, the weather, the altitude and enemy positions and possibly tactics. In a similar sample space, a group level SA of the micro model would be the crew of a C130 airplane; and the active participants communicate their collective SA.

In the above examples, the micro model SA at both the individual and group level demonstrates the close relationship between the participants and the entity (air plane).

Macro model

For the macro model of SA, we note that both individual and group level the participants exercise control over more than one entity or scenario.

In JSAF, at the individual level, a player can control a range of tasks. For example, a blue player can engage in a *mano-a-mano* confrontation with a red player; and in a different setting, a different player can commandeer a battalion of tanks engaging in full-scale combat.

At the group level (within a cell), the players can collaborate their SA through the exchange of information and together meeting a common mission objective. This objective can be for example, a red force eluding the blue force, and vice versa for the blue force to capture the red force.

The proposed taxonomy is consistent with Endsley's view of situation awareness. It further organizes SA roughly in terms of how many entities are controlled by

the player(s). Much research has been focused on the micro model, e.g. aviation and flight simulators. The above example places JSAF in a macro model SA category, and calls on researchers to explore SA and its impact on the meeting mission objectives.

Use of SA to evaluate the effectiveness of sensor technology

Computer generated force simulations represent the real world at the entity level. For example, entities can be humans and vehicles, like aircrafts, ground vehicles and surface vessels; Or, they can be embedded systems, like IFF, radio transmitters and sensors. Or, they can even represent the environment, like fog, precipitation and cloud layers. These simulated entities interact with each other by sending messages. They periodically emit *state* messages reporting their internal state attributes, such as their location, movement, damage state, camouflage state, and so on. Also, they emit *interaction* messages indicating what they did, for example, an aircraft can send a weapon fire message, and a radio transmitter can send a radio signal message. The entities are always truthful in the messages, so the entire set of the state and interaction messages during a simulation defines the *simulation ground truth*.

sensors and human intelligence, to provide them with a *perspective* on the contents of the simulation. Using this perspective the players develop their situation awareness, see Figure 1. With respect to the simulation ground truth, the players' perspective is partial, approximate and delayed. The players do not have enough wherewithals to deploy observers everywhere and all the time. Even if observers were deployed, their observations are not exact. For example, due to the inaccuracies of the sensor technology (as simulated by the observer sensor entity), an observer may misclassify a heavy truck as a tank. Also, their observation message sends to the players maybe delayed. This delay sometimes is an artifact of the underlying computing infrastructure, or sometimes it is inserted on purpose to emulate actual time delays that are consistent with the real world.

Currently, we are participating in the Joint Urban Operations (JUO) Urban Resolve exercises. One of the key objectives of the exercises is to determine in complex urban battle environments the potential effectiveness of proposed 2015 sensor technologies [cite Dehncke's paper]. From the point of view of the Blue force against the Red force, Figure 1 illustrates three potential methods of evaluating the contribution of sensors. One is to compare the Blue force

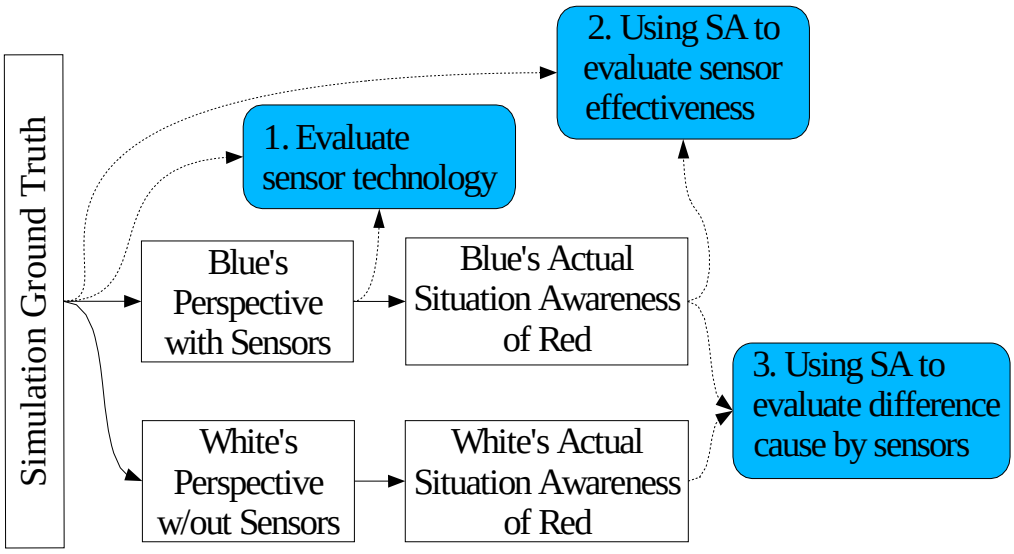


Figure XYZ: Evaluating sensor effectiveness using multiple situation awareness viewpoints.

Typically human simulation players do not directly see the state and interaction messages of the opposing forces, or even all the messages from their own force. The players rely on simulated observer entities, such as

perspectives against the simulation ground truth; two is to compare Blue SA against simulation ground truth; and three is to compare the Blue players' situation awareness against the White cell's situation awareness.

In this case we define that the White cell forms its situation awareness without using the proposed new sensor technologies.

Evaluate Sensor Technology

Method one, perspective versus ground truth, provides an absolute measure of usefulness of the sensor technology. This method utilizes all available data using all the messages from the underlying simulation and all of the sensor data output. Comparing these two types of data, we can determine exactly which entities were detected, and which were not. Of the detected entities we can determine which sensors did the detection, for how long, and if the entities were classified correctly. Of the undetected entities, we can determine if the failure to detect was due to sensor technology, or because no sensors were deployed near the undetected entities. These types of measurements are very useful in determining the usefulness of the sensors and the sensor deployment patterns. Indeed within JFCOM J9 we are developing a range of tools using this method [cite Grabener paper]. Tools we are developed include sensor-target and truth-perception scoreboards. Sensor-target scoreboards indicate which sensor types are more adept at recognizing which entity types. Truth-perception scoreboards indicate the frequency that tracked targets are classified correctly and incorrectly. If targets are classified incorrectly, they indicate the distribution of category types in which the targets are misclassified.

Using SA to evaluate Sensor Effectiveness

Method two, Blue's SA versus ground truth, provides measurements based on how effectively the sensor information is being used. During the exercise the players are typically overloaded with data and with operational tasks that they must perform, such as controlling the entities. In realtime they need to sift through the data, understand it and act upon it. Method one assumes unlimited computing resource and unlimited processing time. For example, generating the scoreboards requires examining Gigabytes of data. The ability of the players to analyze the data is necessarily constrained by the cognitive limits of human memory, attention focus, multi-tasking under workload, and pattern/schema matching ability*[and other limits???...]*. So, in terms of information content Blue's SA is necessarily a *subset* of the information content of Blue perspective. Blue's SA is not a strict set, since the players may misinterpret the information within the Blue's perspective. By comparing the Blue's SA against the ground truth we are able to determine how effectively the sensor data is being used.

Use of Control Group in Assessing Sensor Effectiveness

Method three, Blue's SA versus provides a potentially a fairer way to judge sensor effectiveness. Here we propose a Blue Cell control group, which with the exception of not using the new sensor technologies, are the same as the Blue players. The Control group still receives all the data from traditional observers and sensors, and it still must perform the operational tasks of the Blue players. This measurement helps to quantify the advantages offered by the new sensor technologies, and if the advantage is offset by the extra cognitive load imposed upon the players.

Use of cognitive methods to increase accuracy of SA measurement

Objective Measures

In general, it is much more desirable to obtain objective measures but because of the difficulty in doing so researchers investigating situation awareness tend to rely on subjective measures. Our goal is to develop quantitative measurements of situation awareness for the various levels in our domain. Objective means that may be used for this purpose can be direct experimental techniques, such as probes, and verbal protocols (cite).

Currently, JSAF records player's situation awareness by having them annotate "Situation Awareness Objects" (SAOs) on the computer screen during the exercise. SAOs are pointers that indicate the presence and direction of movement of the opposing force. When the exercise is complete, overall SA is evaluated by comparing the total SAOs recorded against the opposing forces activities. As it stands now, the process is manual and subjective, which leaves room for improvement.

Use of Cognitive Science and SA

Applying what we know about the limitations and biases of cognition can help us increase the validity of our measurements of sensor effectiveness. The analysis is beyond the scope of this paper.

The JSAF exercise has some similarities to some of the cognitive experimental paradigms so we may be able to borrow. We will show how they are similar.

Three Pronged Approach

An experimental approach adapted from research methods in narrative comprehension is the three-pronged approach. The first prong in this approach corresponds to a set of theoretical predictions. The second prong corresponds to the use of verbal protocols/ and or subjective measures. The third prong corresponds to the collection of online behavioral measures. The purpose of the two types of data will be to provide converging evidence for the theoretical predictions of the first prong, somewhat of a different use than in narrative comprehension.

This approach may be used to study situation awareness errors by formulating a set of hypothetical predictions based on intuition or experience about individual and group situation awareness. Subjective evaluation or measurement based on verbal protocols or interviews that are given after the exercise can include: (a) an effectiveness form, (b) individual or collective group discussion or "hot wash", and (c) a third party observer. Objective measurements include the introduction of experimental probes into the exercise, at the group level situation awareness error is the number of situation awareness objects placed by each individual puckers compared between each SA object place and actual Red Force, at the collective/group level situation awareness error is meeting mission objective (normalize to a certain percentage for each player).

Situation Models in Narrative Comprehension

One potential framework that may prove fruitful to look at in researching situation awareness in JSAF comes from research in situation models in cognitive psychology. By comparing how situations are defined within the two paradigms, we can identify general commonalities and differences that may increase our understanding of situation awareness in JSAF.

In cognitive psychology, a narrative comprehension paradigm has been used to study situation models. This typically consists of reading narratives on a sentence-by-sentence basis and answering experimental probe questions. Readers have no background knowledge of what the story is about until they start reading. As they read, the information they encounter can shift along a number of situation defining dimensions. Research using reading time measures has identified 5 dimensions to which readers are sensitive. These are entity, space, time, goal, and causality. In other words, readers construct a situation model that is updated to correspond to changes in the text's situation. Finally, once the story has ended, readers have encoded a "global static summary" of the story, which corresponds to the completed situation model.

Read Comprehension	JSAF Simulation
Read the story - No apriori knowledge of entities - Time can be told out of order - No spatial knowledge - Unknown goals - Causality is "fixed"	Initially Starts the Vignette Some background: - apriori menu of entities - no time shift expected - spatial boundary - 2 sets of goals - causality is dynamic
Information acquired throughout the reading process can cause shift along the five dimensions: - entity - space - time - goals - causality	Information acquisition = the game/experiment: - no time shift - sensors provide space and entity shift - inferential provide goals and causality shift
Completion State Global static summary - character summary - plots - space and time summary	End of Vignette Global static summary - effectiveness of mission - goals evaluated - effectiveness of sensors

Table 2: Comparison between reading comprehension narrative and JSAF simulation along the five cognitive processes dimensions

Although quite different experimentally, the JSAF paradigm can be compared to the narrative comprehension paradigm. Reading is naturally a more passive activity than game playing. Players are provided with a vignette so there is some background knowledge. The knowledge includes information about the situation dimensions, such as an apriori menu of entities and spatial boundaries and geographical constraints. As the game progresses, situation information comes from various sources: entity and spatial information comes from the sensors, whereas goal and causal information is inferred the movements of the entities. Note that the game continues uninterrupted so there is no time shift. When the game is over, the result is a global static summary, analysis of the end result of the game in which the effectiveness of the strategy, the goals of the mission, and the effectiveness of the information provided by the sensors are evaluated. The challenge is to objectively measure situation awareness, probes, and causality shifts.

CONCLUSIONS AND FUTURE WORK

In this paper, we laid the foundation for integrating mental models with physical models in the context of the JSAF experiment. Of specific interest is player's situation awareness. An alternative taxonomy of situation awareness is proposed that positions JSAF in the proper domain of situation awareness. It is argues that cognitive methods play an important role in the measurement of situation awareness and the development of quantitative models in JSAF.

Our future work will focus on developing a prescriptive and descriptive model of situation awareness within the synthetic battlefield arena and incorporating quantitative and qualitative measures into our JSAF experiments. We will use the results to validate sensor technology, and with this commanding officers can train their players and develop doctrine for various engagement tactics.

ACKNOWLEDGEMENTS

This material is based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

.

REFERENCES

Pew, Richard W. and Anne S. Mavor (1997) (ed). *Representing Human Behavior in Military Simulations*. Washington D.C.: National Academy Press.

Endsley, Mica R. and Daniel J. Garland (2000) (ed). *Situation Awareness Analysis and Measurement*. Mahwah, N.J.: Lawrence Erlbaum Associates, Publishers.

Gentner, Dedre and Ablert K. Stevens (1983) (ed). *Mental Models*. Hillsdale, N.J.: Lawrence Erlbaum Associates, Publishers.

Matheus, Christopher J., Mieczyslaw M . Kokar, and Kenneth Baclawski (2003). A Core Ontology for Situation Awareness. *In Proceedings of Sith International Conference on Information Fussion.* Cairns, Australia, pp 545-552.

Graebner, Robert, Gregory Rafuse, Robert P. Miller, and Ke-Thia Yao (2003. The Road to Successful Joint Experimentation Starts at the Data Collection Trail. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlanda, FL.

Williams, Richard and John J. Tran (2003). Supporting Distributed Simulation on Scalable Parallel Processor Systems. . *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlanda, FL.

McCarley, Jason S., Christopher D. Wickens, and William J. Horrey (2002). A computational model of attention/situation awareness. *Proceedings of the 46th Annual meeting of the human factors and ergonomics Society*. Santa Monica, CA.